Finite Difference-Boundary Element Methods in Infinite and Semi-infinite Media in Geomechanics

Volume 1

by

Ziad Halabi

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Civil Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The engineering problems in Geomechanics and Geotechnical fields are commonly treated through the infinite or semi-infinite media. The best approach to solve these problems numerically is by coupling a finite element or a finite difference with boundary element numerical methods. Coupling the bounded domain modelled by Flac^{3D}, a well-known program that implements an explicit finite difference method, with the boundary element method, which satisfies exactly the governing Partial Differential Equations (PDE) in the surrounding infinite or semi-infinite medium, combines the capabilities and the advantages of both methods. The Domain Decomposition Method (DDM) partitions the task of solving the PDE into separate computations over the coupled sub-domains. This method allows the FDM (Flac^{3D} program) and the Boundary Element Method (BEM) program to work independently and interactively. In contrast, at the level of discretized equations, the coupling method requires building a complicated unified system of equations. Therefore, a Domain Decomposition Sequential Dirichlet-Neumann Iterative Coupling Method is developed in this thesis to couple both programs. The method is applied in four cases, 2D and 3D infinite and semi-infinite domains, using the appropriate fundamental solutions in the Boundary Integral Equation required for each case. After applying this method, the mechanical responses computed by Flac^{3D} is corrected and the same responses far from the bounded domain are computed with less computer runtime (CPU) compared with the uncoupled Flac^{3D} solution. The method is also verified by comparing the obtained numerical results with the corresponding analytical solutions. Two BEM pre and post processing intrinsic plug-ins are created, which provide access to the data of Flac^{3D}, as well as the internal structure of the programming language embedded within Flac^{3D} program. These intrinsics are 10 to 100 times faster to execute than the functions created using the Flac^{3D} embedded language. Furthermore, the complementary part of the Kernels is derived based on Mindlin's fundamental solutions. These Kernels are required to compute the stress inside the 3D semi-infinite domain.

Acknowledgments

I am indebted to my supervisor Professor Leo Rothenburg for his patience, guidance, and financial support during my study at the University of Waterloo.

I am also grateful for all the suggestions from the committee members, Professor Cascante Giovanni, Professor Maurice B. Dusseault, Professor Hamid Jahed and Itasca Staff.

Table of Contents

Chapter 1

Introduction

1.1 The Motivation and Objective of the Research

Although most of the problems in Geo mechanics take place either in a semi-infinite medium for close to ground surface problems or in an infinite medium for deep problems, most of the commercial numerical analysis programs do the analysis in the bounded domains. The medium in Geo mechanics is considered in general as an infinite or semi-infinite medium because the space occupied by the body under study is very small compared with the surrounding medium of the ground.

There are number of well-developed and verified numerical analysis programs to address geomechanics problems, such as ADINA, ABAQUS, ANSYS, PLAXIS, FLAC^{3D}, etc. Most of these programs are sufficient to solve geomechanics problems of considerable complexity. However, none of them are based on publically available source codes to allow additional developments to deal with efficiency issues for infinite and semi-infinite media. Nevertheless, FLAC^{3D} provides an unparallel access to all internal operations through an embedded programming language (FISH) and C++ plug-ins. It is for these reasons the work described in this thesis involves development of a plug-ins software and related tools to FLAC^{3D} to treat problems occur in infinite and semi-infinite mediums.

Flac^{3D} program stands for Fast Lagrangian Analysis of Continua in 3 Dimensions. This program is one of the well-known numerical finite difference analysis programs because of the numerous following capabilities and advantages:

- Flac^{3D} has up to twelve constitutive models;
- Conducts fast non linear numerical analysis for elasto-plasticity problems;

- Has two different modes of small infinitesimal strain and large strain modes;
- Analyzes static and dynamic loading systems;
- Solves the problems in a non equilibrium state: Flac^{3D} indicates state of steady plastic flow, equilibrium or non equilibrium states at the end of incremental strain rate-stress analysis;
- Performs mechanical-groundwater flow coupling, thermal-mechanical coupling and thermal-groundwater flow coupling;
- Executes operations sequentially (sequential excavations);
- Produces contours of stresses, strains, displacements and other vectors and tensors;
- Has its own programming language (*FISH*) embedded in commands lines of Flac^{3D} data file;
- Users can create their own C++ *FISH* intrinsics and load them at runtime as plug-ins;
- Introduces structural elements (beams, shells, cables…) and many other features into the model.

All these features make Flac^{3D} a very efficient tool to study and research numerous problems and applications in Geo mechanics, everything from studying loading capacity of slopes and foundations, sequential excavation of intersecting tunnels, mining design, behaviour of oil reservoirs, fully saturated flow and dissipation of drained and undrained loading, to studying dynamic effects of explosive loading and materials.

The finite difference method (FDM) similar to the finite element method (FEM) transforms the governing differential equations into nodal force-displacement-stiffness equations. Flac^{3D}, as a finite difference program, approximates the governing differential equations by discretizing a bounded domain, which completely contains the underground body under study, into a

number of zones divided in turn into one or two layers of constant strain-rate tetrahedra. Obviously, because it is impossible to cover the whole infinite or semi infinite medium of the ground that surrounds the bounded domain by zones, artificial or truncation boundaries are placed in the bounded model and zero displacements or in-situ stresses are imposed as boundary conditions. This will introduce an error into the calculated mechanical response that cannot be deleted unless the truncation boundaries are placed very far from the analyzed body. Expanding the domain needs an increasing number of zones, so more runtime is wasted.

1.2 Summary of the Research Main Contributions

Coupling, the bounded domain modelled by Flac^{3D} with the boundary element method that satisfies exactly the governing DE in the surrounding infinite or semi-infinite medium and approximates the boundary conditions, eliminated the error introduced by the truncation boundary and reduced the runtime by reducing the zone numbers. Furthermore, the mechanical responses very far from Flac^{3D} bounded domain (e.g. at the ground surface) were computed with less cost in runtime. Only the boundary was discretized into boundary elements in BEM which reduced practically the dimensionality of the problem from a 3D to a 2D problem and from a 2D to a 1D problem. This of course explains the great reduction of unknowns (displacements or tractions) to be obtained in BEM compared with the unknowns (displacements or forces) to be obtained in FDM or FEM for a certain problem. In this thesis, the BEM being used establishes a direct relationship between the prescribed and the unknown boundary nodal displacements and tractions utilizing Betti's reciprocal theorem and Somigliana's identity. Providing an improved BE-FD coupling methods to solve problems in infinite and semi-infinite media has been accomplished through the following theoretical and computational developlements:

1. Theoretical developments:

- Adaptation of one of BE-FD or FE coupling methods to specifically Flac^{3D}-BE coupling methods. Hence, A Domain Decomposition Sequential Dirichlet-Neumann iterative and Single-step methods are developed to couple FD with BE methods.
- Further development of Mindlin's analytical solution for point force in the half-space to derive the BIE kernels for computing the stress at any point of the 3D Semi-infinite medium.
- Testing the algorithm analytically on a simple problem for which an analytical solution is available to both finite and infinite cases (Lame's problem of a thick cylinder).

2. Computational developments:

- Application of the developed coupling methods on four cases, 2D and 3D infinite and semi-infinite domains and for elastic and elasto-plastic constitutive models, using the appropriate fundamental solutions in the Boundary Integral Equation required for each case.
- Introducing the effect of the surrounding unbounded domain into the Flac^{3D} bounded sub-domain, corrected the mechanical responses computed independently by Flac^{3D}.
- Coupling bounded Sub-domain analyzed nonlinearly by FLAC^{3D} with unbounded sub-domain analyzed by BEM combined the advantages of both methods and achieved a converging solutions with less run and analysis time.
- Verification of the developed coupling methods by comparing the obtained numerical results with the corresponding analytical solutions.
- Development of two BEM pre and post processing intrinsic C++ *FISH* plug-ins, which provide access to the data of Flac^{3D}, as well as the internal structure of the programming

language embedded within Flac^{3D} program. These intrinsics are 10 to 100 times faster to execute than the functions created using the Flac^{3D} embedded programming language (*FISH*).

Chapter 2

Linear Boundary Element Method

2.1 Introduction and Literature Review

The analytical solution of the governing differential equilibrium equations and boundary conditions in continuum mechanics cannot be found except for limited cases of geometry and loading conditions. Numerical methods, and the Boundary Element as one of these methods, approximate the solution for the problems with more complicated geometry and BCs. In general, numerical methods can be classified into finite difference, finite element, and boundary element methods. The first two methods demand the subdivision of the whole domain under consideration while the boundary element method needs only the boundary of the region to be subdivided.

(i) The first widely known method was finite differences by Southwell [1] in 1946. The governing differential equilibrium equation is computed at the finite difference mesh (FDM) nodes (see Figure 2.1) using a finite difference approximation. The mesh is built so that the used approximation provides an accurate enough overall solution [2]. The advantages of FDM are its generality; it can be applied over diverse spectrum of problems, deal easily with non homogeneity, and need no numerical integration. One of the disadvantages of the method is its inaccurate modelling of infinite medium problems.

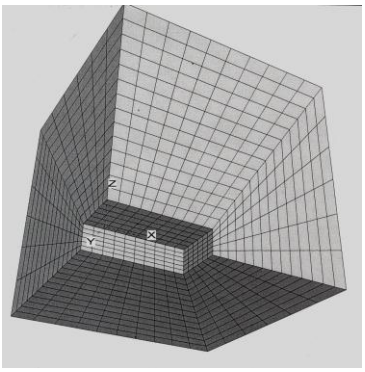


Figure2.1 Finite Difference Mesh

(ii) The finite element method is based on the original idea of Ritz [11] in 1909 and developed by Zienkiewicz [12] in 1977, Owen and Hinton [13] and others. The solution precisely satisfies the boundary conditions, and the error in satisfying the governing differential equilibrium equation is minimized. To compute the response due to prescribed boundary conditions or loadings, functions interpolate the variable in each subdivision or element (as long as it is impossible to have a function cover the whole domain and satisfy BCs), are chosen using residual weighted methods. This method minimizes the error of violating the differential equilibrium equation at every point of the domain [3]. A three dimensional finite element mesh for spherical excavation is shown in Figure 2.2. Tens of thousands of degrees of freedom are prescribed and computed; this consumes a long runtime using a personal computer. The domain is either infinite or semi infinite but truncation or an artificial boundary should be introduced imposing either zero displacement at the boundary or in situ stress, with an error included in the problem modelling from the very beginning. The advantages of FEM are its generality, suitability to problems with finite domains, applicability to inhomogeneous material properties and non-linear behaviour. Moreover, FEM requires simple numerical integration, models boundaries and BCs well, and mixes a broad range of elements. The disadvantage of this method is it models infinite and semi-medium problems with less accurate output.

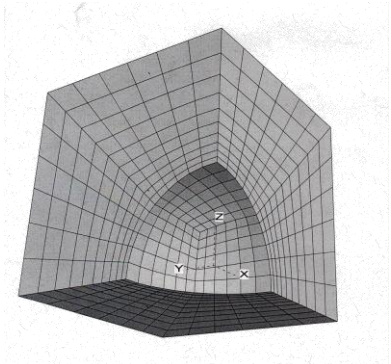


Figure2.2 Finite Element Mesh

(iii) The boundary element method (BEM): its roots can be traced back to the theory of integral equations by Fredholm [14] in1905 and developed later by Mikhlin [15]. In this method a different approach was proposed by Trefftz [16]. The solutions exactly satisfy everywhere in the domain, the governing differential equilibrium equation using certain functions, whereas approximating the boundary conditions with minimized error. No need to subdivide the domain into elements if the chosen functions satisfy exactly the differential equilibrium equation, but the error in satisfying boundary conditions is minimized using Boundary Integral. The integral is numerically evaluated by subdividing the boundary into elements in which tractions or displacements are interpolated using functions similar to those used in FEM [3]. In Figure 2.3, the boundary element mesh is shown (spherical excavation surface) with less than 500 degrees of freedom and a more accurate solution for the stress is obtained inside the domain as long the method exactly/precisely satisfies the differential equation. The BEM method reduces the dimensionality of the problem (surface instead of volume), and easily deals with the infinite medium. The disadvantage of BEM is that the functions or the fundamental solutions that satisfy the DE are singular and require special care in integrating them over the boundary [3].

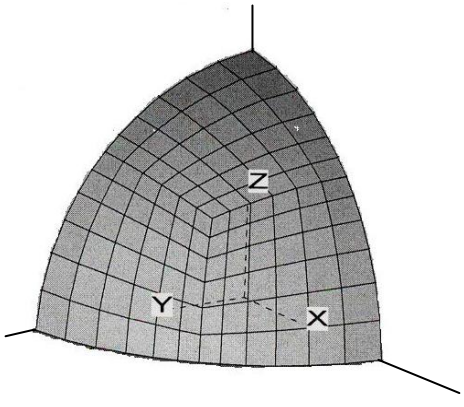


Figure2.3 Boundary Element Mesh

The boundary element method is classified into two main methods, the direct and the indirect boundary element methods:

I- The indirect boundary element method: the original method proposed by Trefftz who applied fictitious forces or sources in points inside the domain, computed the density of these sources and in the next step used them to calculate the unknown boundary values. The disadvantage of this solution is that it is inaccurate, difficult to converge, and to computerize for a general problem (case specific). Different kinds of indirect boundary element methods were developed by Salamon [17], Massonet [18], Plewman et al. [19], Saterfield and Crouch [20] and Banerjee and Driscoll [21] such as the fictitious stress method and displacement discontinuity method. In these methods, fictitious density functions, with no direct physical meaning, and assumed to be distributed over the domain boundary are obtained and used in the following step to compute the unknown boundary values. These methods used by early researchers are less popular nowadays than the direct boundary method. Most probably because they need theoretical integration of the fundamental solutions over the boundary elements, and they are more computation demanding than the direct boundary method.

II- The direct boundary element method: Originated for elasto static problems in Rizzo's [22] work. Solutions for three dimensional problems were obtained by Cruse [23] in 1969. Unknown boundary values in this method are obtained directly (no need for fictitious sources) using the reciprocal or Betti theorem. The boundary integral equations based on this theorem are written in terms of displacement, traction, body forces, and fundamental solutions. More will be detailed and discussed about this method in the following sections of this research.

2.2 Theoretical Background in the Direct Boundary Element Method

Since the functions mentioned before (Green's functions) should satisfy exactly the differential equations, solutions of these equations must be reached as simple as possible. One of these solutions is Kelvin's problem solution based on concentrated force applied in infinite medium derived by W. Thomson or Lord Kelvin [24] (1848). For a load applied in a semi-infinite medium problem (free traction surface), Melan's [25] solution is the fundamental solution in two dimensional domains. Mindlin's solution is the fundamental solution in three dimensional domains.

The following two paragraphs are a summary of the thorough and well explained derivations of Kelvin's fundamental solution and Somigliana's Identity presented by J. Kane [2].

2.2.1 Kelvin's Fundamental Solution

In an isotropic medium, the 15 unknowns present in linear elastic problems are related by the following differential equations written in a tensor notation as the following:

Strain - Displacement: $\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i})$ (2.1)

Stress - Strain: $\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2\mu\varepsilon_{ij}$ (2.2)

Differential Equilibrium Equations: $\sigma_{ji,j} + b_i = 0$ (2.3)

These equations can be written in terms of displacements and get Navier's equations of elasticity:

$\mu u_{i,jj} + (\mu + \lambda)u_{j,ji} + b_i = 0$ *or in vector form:* $\mu\nabla^2\mathbf{u} + (\mu + \lambda)\nabla(\nabla.\mathbf{u}) + \mathbf{b} = 0$ (2.4)

Note: vectors or tensors are written in bold font. **u** is the displacement vector; **σ** and **ε** are the stress and the strain tensors, respectively. λ and μ are Lame's constants given by:

$$\lambda = \frac{\nu E}{(1-2\nu)(1+\nu)}; \mu = G = \frac{E}{2(1+\nu)} \quad (2.5)$$

E is modulus of elasticity, ν is Poisson's ratio and G is shear modulus, **b** is body forces defined as forces per unit volume.

Navier's three equations are coupled. This means that every equation has the three unknown displacements $u_1 = u_x$, $u_2 = u_y$, and $u_3 = u_z$ at the same time. The analytical solution for these equations is very difficult to reach, so some techniques were used to decouple them, such as the Galerkin vector approach given by:

$$2\mu u_i = 2(1-\nu)g_{i,jj} - g_{j,ji} \tag{2.6}$$

If **g** in the Galerkin vector given by equation (2.6) is substituted in equation (2.4), the equilibrium equation in terms of the Galerkin vector becomes:

$$(1-\nu)g_{i,kkjj} + b_i = 0 \quad \text{or in vector form} \quad \nabla^2(\nabla^2\mathbf{g}) + \frac{1}{(1-\nu)}\mathbf{b} = 0 \tag{2.7}$$

By substituting equation (2.1) and equation (2.6) in equation (2.2), the stress components in terms of the Galerkin vector are given as the following:

$$\sigma_{ij} = (1-\nu)(g_{i,kkj} + g_{j,kki}) - g_{k,kij} + \nu\delta_{ij}g_{l,lkk} \tag{2.8}$$

The force applied in Kelvin's problem in a 3D elastic infinite medium can be considered as the limiting case of body force **b** acting on a sphere as the radius of the sphere shrunk to zero. The intensity of the body force in this sphere is increased by decreasing its radius such that the resultant load **P** remains constant."This definition is consistent with the definition of dirac delta function [see Figure 2.4]. This function is configured so that its integral is equal unity over any domain where the function turns on" [2].

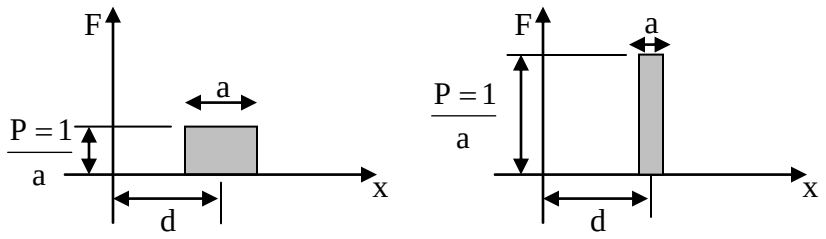


Figure 2.4 Unit Rectangular Pulse Function

$$F(x,d,a)=\begin{Bmatrix} 0 & ; & x<d-\dfrac{a}{2} \\ \dfrac{1}{a} & ; & d-\dfrac{a}{2}\le x\le d+\dfrac{a}{2} \\ 0 & ; & x>d+\dfrac{a}{2} \end{Bmatrix}$$

$$\text{Dirac}-\text{delta}:\delta(x-d)=\lim_{a\to 0}F(x,d,a)$$

The fundamental solutions are obtained by applying a load at each direction in turn (as an example z direction shown in Figure 2.5) in the center of a sphere. The traction vector acting on the surface of the sphere is obtained to be in equilibrium with the applied load as the following:

$$\Sigma F_3 = P+\int_S t_3 dS = 0 \tag{2.9}$$

The integration in the above equation is a surface integration done over surface S.

Using the Cauchy rule ($t_i=\sigma_{ji}n_j$), equation (2.9) becomes:

$$P+\int_S \sigma_{j3}n_j dS = P+\int_S \sigma_{j3}\frac{x_j}{r}dS = 0 \tag{2.10}$$

n is the normal vector on the sphere surface.

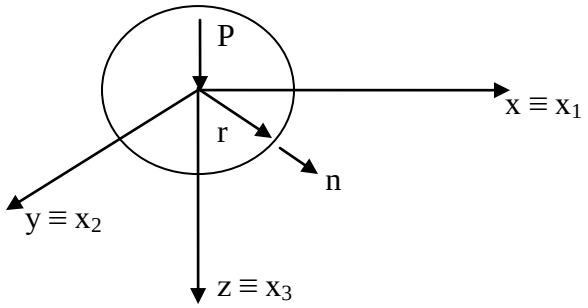


Figure 2.5 Kelvin's Solution Sphere

If the stress component σ_{j3} is obtained from equation (2.8) and substituted in equation (2.10), in terms of the Galerkin vector, this equation will be as the following:

$$P + \int_S r^{-1} x_j \{(1-\nu)[g_{3,kkj} + g_{j,kk3}] - g_{k,k3j} + \nu\delta_{3j} g_{l,lkk}\} dS = 0 \tag{2.11}$$

The integral in the above equation should always be true, independent of the size of the sphere or the value of its radius r. This fact helps to speculate the form of the Galerkin vector as the following:

$$g_1 = g_2 = 0;\ g_3 = f.\ r \tag{2.12}$$

Where: f is constant. Derivatives of **r** to the third order are given by:

$$r_{,i} = x_i / r$$

$$r_{,ij} = \frac{\delta_{ij} - r_{,i} r_{,j}}{r} \tag{2.13}$$

$$r_{,ijk} = \frac{[3r_{,i} r_{,j} r_{,k} - (r_{,i}\delta_{ik} + r_{,i}\delta_{ki} + r_{,k}\delta_{ij})]}{r^2}$$

Derivatives of **g** can be easily obtained by deriving equations (2.12) and using equations (2.13). Using the polar coordinate system and substituting the derivations of r and g in equation (2.11) gives us the following equation independent of r:

$$-P = 8\pi f\,(\nu - 1) \tag{2.14}$$

If equation (2.12) is substituted in equation (2.6), we get the following:

$$2\mu u_i = f\{2(1-\nu)\delta_{3i} r_{,jj} - r_{,i3}\} \tag{2.15}$$

In a very similar procedure, similar equations are obtained by applying the load in the other two directions, x and y. The general equation is written by adding another free index (k) to get the displacement in (i) due to the load applied in (k) direction:

$$2\mu u_{ki} = f\{2(1-\nu)\delta_{ki} r_{,jj} - r_{,ik}\} \tag{2.16}$$

The final form of Kelvin's fundamental displacement solution is obtained by substituting (2.14) and the derivations of **r** into equation (2.16):

$$u_{ki} = \frac{P_k}{16\mu\pi(1-\nu)r}\{(3-4\nu)\delta_{ki} + r_{,k}r_{,i}\} \tag{2.17}$$

The same can be obtained for the stress components by substituting (2.14) and the derivations of **g** into equation (2.8):

$$\sigma_{ijk} = \frac{P_k(1-2\nu)}{8\pi(1-\nu)r^2}(\delta_{ij}r_{,k} - \delta_{ik}r_{,i} - \delta_{ki}r_{,j} - \frac{3}{1-2\nu}r_{,i}r_{,j}r_{,k}) \tag{2.18}$$

Using the Cauchy rule, the fundamental solution for the traction components becomes:

$$t_{ki} = \sigma_{ijk}n_j = \frac{-P_k}{8\pi(1-\nu)r^2}\{r_{,j}n_j[(1-2\nu)\delta_{ik} + 3r_{,k}r_{,i}] - (1-2\nu)[r_{,k}n_i - r_{,i}n_k]\} \tag{2.19}$$

2.2.2 The Reciprocal Theorem and Somigliana's Identity

Betti's reciprocal theorem states that if a linearly elastic body is exposed to two different systems of body and surface forces, the work done by forces of the first system along the displacements of the second system is equal to the work done by the forces of the second system along the displacements of the first system.

A brief proof of Betti's theorem is explained here using the principle of virtual work as the following: Integral of equilibrium equation over the domain (volume) V is equal zero if the bracketed quantity in equation (2.20) equals zero according to equation (2.3). If this integral is multiplied by weighing function (w), the product still equals zero for the same mentioned reason:

$$\int_v (\sigma_{ji,j} + b_i)w_i dV = 0 \tag{2.20}$$

$$\int_v (\sigma_{ji,j}w_i + b_i w_i)dV = \int_v ([\sigma_{ji}w_i]_{,j} - \sigma_{ji}w_{i,j} + b_i w_i)dV = 0$$

Divergence theorem can be used to transform the first integral term from volume to a surface integral: $$\int_S [\sigma_{ji} w_i] n_j dS - \int_V \sigma_{ji} w_{i,j} dV + \int_V b_i w_i dV = 0$$

Using the Cauchy rule gives us the surface integral in terms of tractions:

$$\int_S t_i w_i dS + \int_V b_i w_i dV = \int_V \sigma_{ji} w_{i,j} dV \tag{2.21}$$

The tensor $w_{i,j}$ can be written as sum of its two symmetric and anti-symmetric parts. The product of symmetric tensor σ_{ij} with anti-symmetric part of $w_{i,j}$ equals zero:

$$\frac{1}{2}(w_{i,j} - w_{j,i})\sigma_{ij} = 0$$

This makes the equation (2.21) take the form:

$$\int_S t_i w_i dS + \int_V b_i w_i dV = \int_V \sigma_{ji} \frac{1}{2}(w_{i,j} + w_{i,j}) dV \tag{2.22}$$

The equation is true regardless of the nature of the arbitrary weighing function w_i, which can be taken as a virtual displacement $\bar{\mathbf{u}}$. Traction, body force and stress product virtual displacement is called virtual work; thus, the following is called the principal of virtual work:

$$\int_S t_i \bar{u}_i dS + \int_V b_i \bar{u}_i dV = \int_v \sigma_{ji} \bar{\varepsilon}_{ij} dV \tag{2.23}$$

If the object with domain V is subjected to two systems of loads and deformations and both cases are in equilibrium, the principal of virtual work can be applied on/to both systems. Virtual displacement is an arbitrary function that can be replaced by other kinematic quantities if the equilibrium is still secured:

$$\int_S t_i^{(1)} u_i^{(2)} dS + \int_V b_i^{(1)} u_i^{(2)} dV = \int_v \sigma_{ji}^{(1)} \varepsilon_{ij}^{(2)} dV \tag{2.25}$$

$$\int_S t_i^{(2)} u_i^{(1)} dS + \int_V b_i^{(2)} u_i^{(1)} dV = \int_v \sigma_{ji}^{(2)} \varepsilon_{ij}^{(1)} dV \tag{2.26}$$

For isotropic linear elastic material we have: $\sigma_{ij} = C_{ijmn}\varepsilon_{ij}$ and the constitutive tensor C_{ijmn} is symmetric, which leads us to the conclusion: $\overset{(2)}{\sigma}_{ji}\ \overset{(1)}{\varepsilon}_{ij} = \overset{(1)}{\sigma}_{ji}\ \overset{(2)}{\varepsilon}_{ij}$ (2.27)

If equation (2.27) is substituted into equations (2.25) and (2.26), we will have Betti's reciprocal theorem in the following mathematical expression:

$$\int_S \overset{(1)}{t}_i\ \overset{(2)}{u}_i\, dS + \int_V \overset{(1)}{b}_i\ \overset{(2)}{u}_i dV = \int_S \overset{(2)}{t}_i\ \overset{(1)}{u}_i\, dS + \int_V \overset{(2)}{b}_i\ \overset{(1)}{u}_i dV \tag{2.28}$$

If system (2) is assumed to be the fundamental solution of Kelvin's problem and knowing (see Figure4) that the concentrated force in Kelvin's problem was assumed to be applied as a body force in the form of a dirac delta function, equation (2.28) becomes:

$$\int_S U_{ki}t_i dS + \int_V U_{ki}b_i dV = \int_S T_{ki}u_i dS + \int_V \delta_{ki}(\mathbf{x}-\mathbf{d})u_i dV \tag{2.29}$$

$\delta_{ki}(\mathbf{x}-\mathbf{d})$**:** dirac delta function, **d:** position vector of load **P=** $\begin{Bmatrix} P_1 \\ P_2 \\ P_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}$ and **x** is position vector of field point Q. The fundamental solution is written in upper case letters. Traction and displacement written in lower case letters are the prescribed and desired boundary values. "Selection property of the dirac delta function states that the product of the delta function and another function, integrated over a domain in which the delta function turns on, is equal to the value of the function at the place where the delta function turns on"[2]. This property turns equation (2.29) into:

$$\int_S U_{ki}t_i dS + \int_V U_{ki}b_i dV = \int_S T_{ki}u_i dS + u_k(\mathbf{d}) \tag{2.30}$$

If the body force can be neglected in some problems, the above equation will give us what is called Somigliana's identity:

$$u_k(\mathbf{d}) = \int_S U_{ki} t_i dS - \int_S T_{ki} u_i dS \tag{2.31}$$

The following paragraphs (2.2.3 to 2.2.7) outline G. Beer's [3] *significant elaboration about the numerical computation of boundary integrals and some of its techniques.*

2.2.3 Boundary Integral Equations

Somigliana's identity, based on reciprocal theorem of Betti, built a direct relationship between the prescribed and desired or unknown boundary values as explained above. Boundary integrals that appear in Somigliana's identity are computed numerically by placing the unit load points P inside the domain and the field points Q on the domain boundary (see Figure 2.6 and Figure 2.7). The solution will converge if the number of points Q and P are increased.

Equation (2.31) using matrices notation, preferred by engineers because of its simplicity and vector format might be rewritten as the following:

$$\mathbf{u}(P) = \int_S \mathbf{U}(P,Q)\mathbf{t}(Q)dS - \int_S \mathbf{T}(P,Q)\mathbf{u}(Q)dS \tag{2.32}$$

Where for three-dimensional elasticity problems:

$$\mathbf{u}(Q) = \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix} \quad , \qquad \mathbf{U}(P,Q) = \begin{bmatrix} U_{xx} & U_{xy} & U_{xz} \\ U_{yx} & U_{yy} & U_{yz} \\ U_{zx} & U_{zy} & U_{zz} \end{bmatrix} \quad \text{Symmetric tensor}$$

$$\mathbf{t}(Q) = \begin{Bmatrix} t_x \\ t_y \\ t_z \end{Bmatrix} \quad , \qquad \mathbf{T}(P,Q) = \begin{bmatrix} T_{xx} & T_{xy} & T_{xz} \\ T_{yx} & T_{yy} & T_{yz} \\ T_{zx} & T_{zy} & T_{zz} \end{bmatrix} \quad \text{Non Symmetric tensor}$$

The domain shown in Figures 2.6 and 2.7 is infinite. Both systems of tractions and displacements, system (1) as the problem to be solved and system (2) as the fundamental solution, are applied on a closed arbitrary contour represented by a dotted line in the figures. Stress or tractions are shown on the boundary of the continuum by cutting through the dotted

line. The continuum is divided by the cut, depending on the outward normal vector **n** of the boundary, into two parts: internal finite and external infinite domains (see Figure 2.8).

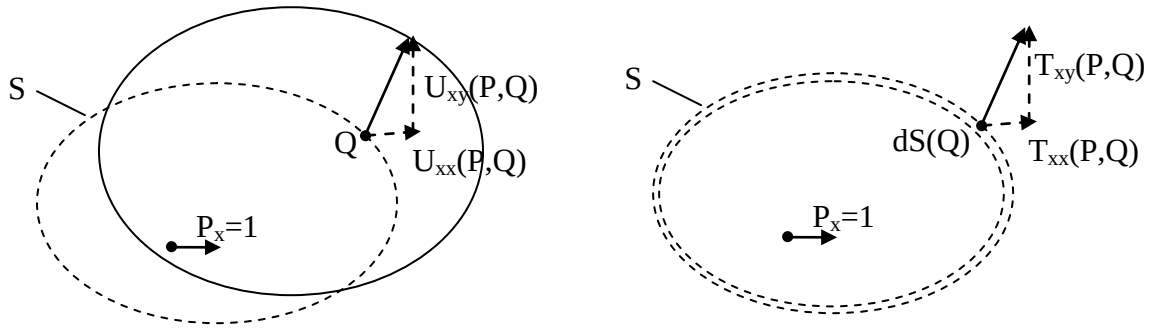


Figure 2.6 Two-Dimensional Kelvin's Fundamental Solutions (system 2) Applied over a Closed Boundary

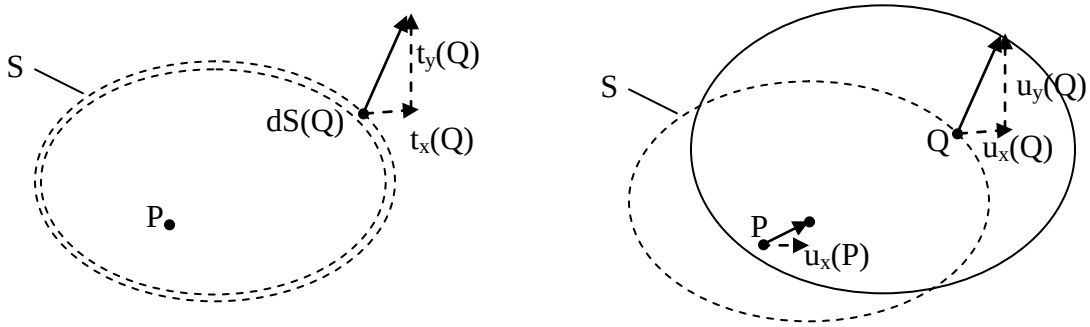


Figure 2.7 Two-Dimensional Prescribed and Unknown Boundary Values (system 1) Applied over a Closed Boundary

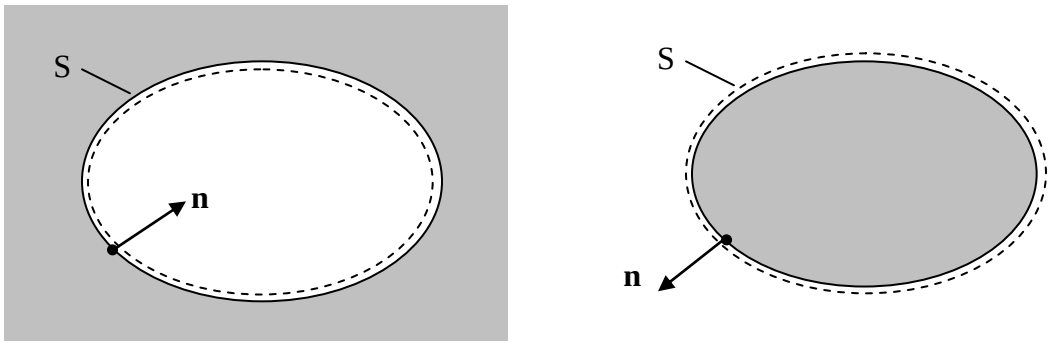


Figure 2.8 Exterior Infinite Domain (left) and Interior Finite Domain (right)

2.2.3.1 The Cauchy Principal Value

In a direct method it is no longer necessary to have fictitious load points as introduced by the Trefftz method, but it is still required to have additional two sets of load P and field Q points. The method will be easier to implement and faster on the computer if the load field points are placed both on the boundary. The problem now is that because of the singularity of the fundamental solution, U varies in 3D problems according to 1/r known as weakly singular and T varies according to $1/r^2$ known as strongly singular. Some integrals in equation (2.32) "only exist in the sense of limiting value as P approaches Q" [3].

Exclusion for 3-D elastostatic problems is defined around load point P. The exclusion is a hemispherical bump centered at P with radius ε as shown in Figure 2.9. Integrals in equation (2.32) will be split into integrals over $S\text{-}S_\varepsilon$: the part of the boundary surface without the bump zone, and s_ε, the bump surface. If ε is taken to zero no difference if we integrate over s_ε or if the integration is done over S_ε. Breaking up the integral into two surface integrals─as the limit of the surface around the singularity, approaches zero─ is called the integral in the Cauchy principal value sense.

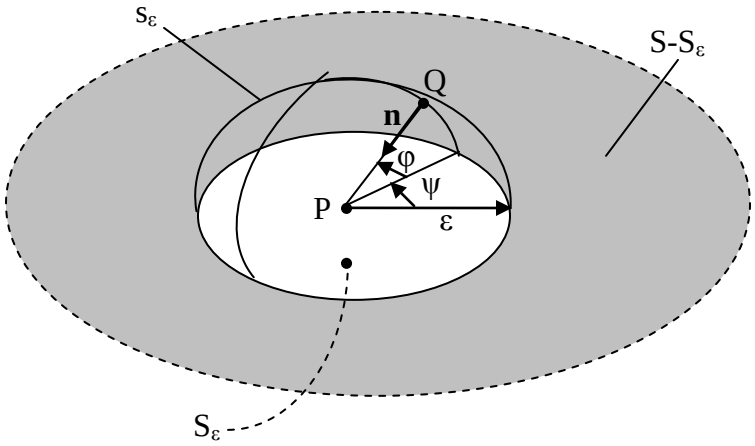


Figure 2.9 Integral over Hemispherical Bump Surface in the CPV Sense

Right hand side of equation (2.32) can be written now as:

$$\int_S \mathbf{U}(P,Q)\mathbf{t}(Q)dS - \int_S \mathbf{T}(P,Q)\mathbf{u}(Q)dS = \int_{S-S_\varepsilon} \mathbf{U}(P,Q)\mathbf{t}(Q)dS + \int_{s_\varepsilon} \mathbf{U}(P,Q)\mathbf{t}(Q)dS - \int_{S-S_\varepsilon} \mathbf{T}(P,Q)\mathbf{u}(Q)dS - \int_{s_\varepsilon} \mathbf{T}(P,Q)\mathbf{u}(Q)dS \quad (2.33)$$

For a smooth surface at P, using polar coordinates ψ, φ, and ε as shown in Figure 2.9, the integration limits will be changed into 0 to 2π for the first integral and 0 to π for the second integral (2-D integrals). The second integral in equation (2.32) equals zero as the radius approaches zero:

$$\lim_{\varepsilon\to 0} \int_{s_\varepsilon} \mathbf{U}(P,Q)\mathbf{t}(Q)dS = 0 \quad (2.34)$$

The last integral will be:

$$\lim_{\varepsilon\to 0} \int_{s_\varepsilon} \mathbf{T}(P,Q)\mathbf{u}(Q)dS = -\frac{1}{2}\mathbf{u}(P) \quad (2.35)$$

In both of the above integrals it is assumed that **t**(Q) = **t**(P) and **u**(Q) = **u**(P) as ε in the limit approaches zero and P approaches Q. By substituting equations (2.34) and (2.35) into equation (2.33), the resulting integrals are substituted into eq. (2.32):

$$\frac{1}{2}\mathbf{u}(P) = \lim_{\varepsilon\to 0}\left[\int_{S-S_\varepsilon} \mathbf{U}(P,Q)\mathbf{t}(Q)dS - \int_{S-S_\varepsilon} \mathbf{T}(P,Q)\mathbf{u}(Q)dS\right] \quad (2.36)$$

For a general case where the surface at P may not be smooth eq. (2.36) takes the form:

$$\mathbf{cu}(P) = \lim_{\varepsilon\to 0}\left[\int_{S-S_\varepsilon} \mathbf{U}(P,Q)\mathbf{t}(Q)dS - \int_{S-S_\varepsilon} \mathbf{T}(P,Q)\mathbf{u}(Q)dS\right] \quad (2.37)$$

c is the free term. Obtaining this term will be discussed later in this chapter.

2.2.4 Numerical Computations of Boundary Integrals

Theoretically, if the boundary conditions are satisfied exactly at all points on the boundary, an infinite number of points P = Q are required. But practically, boundary integral equations can be solved using numerical techniques. A numerical solution will be done either by satisfying the boundary conditions at a limited number of Q, when points Q increase until the solution converges, or by setting a norm of the error in satisfying the boundary conditions and minimizing this error, using weighted residuals, over the boundary. The boundary in a numerical solution is divided into boundary elements. The integral equations are written as a sum of integrals over these elements.

2.2.4.1 Discretization of Integral Equations Using Isoparametric Elements

In early beginnings of BIM, in 2-D problems, line elements with constant boundary values were used (known and unknown tractions and displacements are assumed to be constant over the element). The integrals over these types of elements could be derived theoretically. Lachat and Watson [26] introduced isoparametric elements as higher order elements into BEM, which were first used in FEM in 1968. The integral over these elements is computed numerically by the Gauss method (and other methods as well). Theoretical integration was not needed any more and more accurate solution with fewer and higher order boundary elements for more complicated 3D problems could be achieved.

The Elements that have the same functions N_n which interpolate the element geometry (shape functions) and interpolate displacements and tractions inside the element are called isoparametric elements. Geometry, displacements, and tractions inside two dimensional isoparametric elements (for 3D problems) are given in terms of the nodal values by the following:

$$\mathbf{x}(\xi,\eta)=\sum_{n=1}^{N} N_n(\xi,\eta)\mathbf{x}_n^e \;,\; \mathbf{u}(\xi,\eta)=\sum_{n=1}^{N} N_n(\xi,\eta)\mathbf{u}_n^e \text{ and } \quad \mathbf{t}(\xi,\eta)=\sum_{n=1}^{N} N_n(\xi,\eta)\mathbf{t}_n^e \tag{2.38}$$

x, **u**, and **t** are position, displacement and traction vectors for points inside element e.

ξ, η are intrinsic local dimensionless coordinates. The coordinates vary between 1 to -1. N is the element's node number.

$\mathbf{x}_n^e, \mathbf{u}_n^e, \mathbf{t}_n^e$ are nodal position, displacement and traction vectors for element e.

Nodes in these vectors are numbered locally while in global vectors they are numbered globally according to connectivity of the elements.

Two types of isoparametric 2D quadrilateral elements are used in the applications done in this thesis for reasons that will be explained later. The first are quadrilateral serendipity elements with bilinear functions (see Figure2.10-a) given by:

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\xi)(1-\eta) \qquad & N_3 &= \frac{1}{4}(1+\xi)(1+\eta) \\ N_2 &= \frac{1}{4}(1+\xi)(1-\eta) \qquad & N_4 &= \frac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{2.39}$$

The second are quadrilateral serendipity elements with quadratic higher order functions (see Figure2.10-b) given by:

$$
\begin{aligned}
N_1 &= \frac{1}{4}(1-\xi)(1-\eta) - \frac{1}{2}N_5 - \frac{1}{2}N_8 & N_5 &= \frac{1}{2}(1-\xi^2)(1-\eta)\\
N_2 &= \frac{1}{4}(1+\xi)(1-\eta) - \frac{1}{2}N_5 - \frac{1}{2}N_6 & N_6 &= \frac{1}{2}(1-\eta^2)(1+\xi)\\
N_3 &= \frac{1}{4}(1+\xi)(1+\eta) - \frac{1}{2}N_6 - \frac{1}{2}N_7 & N_7 &= \frac{1}{2}(1-\xi^2)(1+\eta)\\
N_4 &= \frac{1}{4}(1-\xi)(1+\eta) - \frac{1}{2}N_7 - \frac{1}{2}N_8 & N_8 &= \frac{1}{2}(1-\eta^2)(1-\xi)
\end{aligned}
\qquad (2.40)
$$

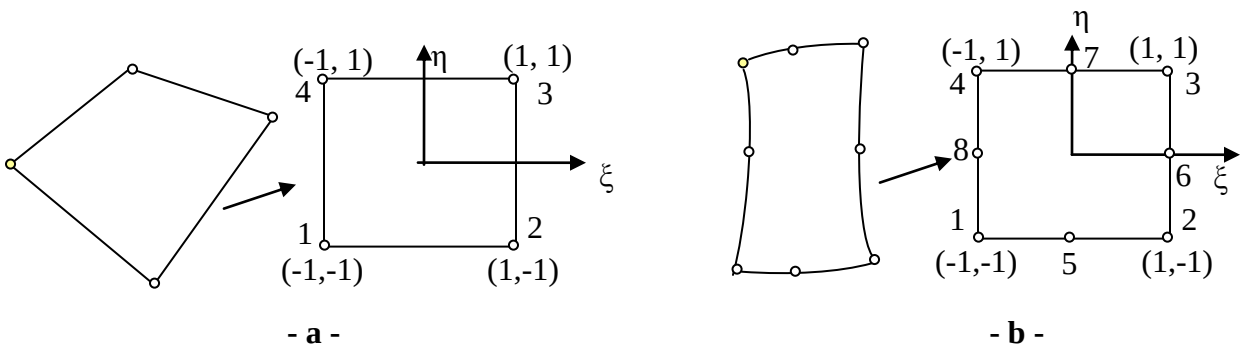


Figure2.10 Quadrilateral Elements; a-Bilinear and-b-Quadratic

If equations (2.38) are substituted into equation (2.37) and the integrals are divided into sum of integrals done over the elements:

$$
\mathbf{cu}(P_i) = \sum_{e=1}^{E}\sum_{n=1}^{N}\left(\int_{S_e} N_n(\xi,\eta)\mathbf{U}(P_i,\xi,\eta)dS(\xi,\eta)\right)\mathbf{t}_n^e - \sum_{e=1}^{E}\sum_{n=1}^{N}\left(\int_{S_e} N_n(\xi,\eta)\mathbf{T}(P_i,\xi,\eta)dS(\xi,\eta)\right)\mathbf{u}_n^e \qquad (2.41)
$$

i= 1, 2…I. I is the total number of nodes on the boundary.

e= 1, 2…E. E is the total number of the boundary subdivisions or elements.

The limit is omitted from the above equation assuming that it exists there implicitly. Nodal displacement and traction vectors are taken out of the integrals because they are constants with respect to the integrations. A limited number of load points have been chosen on boundary P_i

called collocation points at which Betti's theorem is satisfied. Because in 3-D elasticity problems each node has three degrees of freedom each P_i require three equations and the total number of equations is 3I to compute 3I unknowns of boundary values. Either displacement or traction is unknown at each collocation point P_i. Equation (2.41) can be rewritten in the discretized form as:

$$\mathbf{cu}(P_i) + \sum_{e=1}^{E}\sum_{n=1}^{N} \Delta\mathbf{T}_{ni}^{e}\mathbf{u}_{n}^{e} = \sum_{e=1}^{E}\sum_{n=1}^{N} \Delta\mathbf{U}_{ni}^{e}\mathbf{t}_{n}^{e} \tag{2.42}$$

$\Delta\mathbf{U}_{ni}^{e}$ and $\Delta\mathbf{T}_{ni}^{e}$ are the integrations of Kernel interpolation function products over the element area S_e, given by:

$$\begin{aligned} \Delta\mathbf{U}_{ni}^{e} &= \int_{S_e} N_n(\xi,\eta)\mathbf{U}(P_i,\xi,\eta)dS(\xi,\eta) \\ \Delta\mathbf{T}_{ni}^{e} &= \int_{S_e} N_n(\xi,\eta)\mathbf{T}(P_i,\xi,\eta)dS(\xi,\eta) \end{aligned} \tag{2.43}$$

Each of $\Delta\mathbf{U}_{ni}^{e}$ and $\Delta\mathbf{T}_{ni}^{e}$ are stored in matrices for each element of dimension 3N x 3I:

$$[\Delta T]^{e} = \begin{bmatrix} \Delta T_{xx11} & \Delta T_{xy11} & \Delta T_{xz11} & \Delta T_{xx21} & \Delta T_{xy21} & \Delta T_{xz21} & -- \\ \Delta T_{yx11} & \Delta T_{yy11} & \Delta T_{yz11} & \Delta T_{yx21} & \Delta T_{yy21} & \Delta T_{yz21} & -- \\ \Delta T_{zx11} & \Delta T_{zy11} & \Delta T_{zz11} & \Delta T_{zx21} & \Delta T_{zy21} & \Delta T_{zz11} & -- \\ \Delta T_{xx12} & \Delta T_{xy12} & \Delta T_{xz12} & \Delta T_{xx22} & \Delta T_{xy22} & \Delta T_{xz22} & -- \\ \Delta T_{yx12} & \Delta T_{yy12} & \Delta T_{yz12} & \Delta T_{yx22} & \Delta T_{yy22} & \Delta T_{yz22} & -- \\ \Delta T_{zx12} & \Delta T_{zy12} & \Delta T_{zz12} & \Delta T_{zx22} & \Delta T_{zy22} & \Delta T_{zz22} & -- \\ -- & -- & -- & -- & -- & -- & -- \end{bmatrix} \tag{2.44}$$

Element nodes →

↓ Collocation points

Element node No.(n)

Collocation point No.(i)

ΔT_{xy21}

Load direction

Traction direction

2.2.4.2 Computing the Cauchy Principal Value of Strongly Singular Integral with the Free Term (c) Using the Rigid Body Motion Method

Although integration of displacement kernel **U** (weakly singular o(1/r)), shape function product when P_i coincides with Q is a singular integral, it can be obtained numerically as will be shown. But integration of traction kernel **T** (strongly singular $o(1/r^2)$) shape function product in the sense of the Cauchy principal value cannot be found numerically using the regular Gauss Quadrature method. Two different methods are used to obtain these integrals, the first method is developed by Guiggiani and Casalini (will be discussed later for the semi infinite medium) and the second is the concept of body motion by which the free term **c** is obtained too. Rigid body motion implies that if an elastic domain moved by a pure rigid translation, no change of the body shape will occur, consequently this means that the applied tractions are zero. Equation (2.42) can be rewritten as:

$$\mathbf{cu}(P_i)+\sum_{e=1}^{E}\sum_{\substack{n=1\\ g(n)=i}}^{N}\Delta\mathbf{T}_{ni}^{e}\mathbf{u}_{n}^{e}+\sum_{e=1}^{E}\sum_{\substack{n=1\\ g(n)\neq i}}^{N}\Delta\mathbf{T}_{ni}^{e}\mathbf{u}_{n}^{e}=\sum_{e=1}^{E}\sum_{n=1}^{N}\Delta\mathbf{U}_{ni}^{e}\mathbf{t}_{n}^{e} \tag{2.45}$$

Strong singularity is produced when P_i is located in the element (region of integration) coinciding with one of its nodes locally numbered as n and globally as g(n) = i. To have a rigid body translation in x direction a unit displacement u_x=1 and zero in the other directions u_y= u_z= 0 are applied while applied tractions equal zero:

$$\mathbf{c}(P_i)\begin{Bmatrix}1\\0\\0\end{Bmatrix}+\sum_{e=1}^{E}\sum_{\substack{n=1\\ g(n)=i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}1\\0\\0\end{Bmatrix}=-\sum_{e=1}^{E}\sum_{\substack{n=1\\ g(n)\neq i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}1\\0\\0\end{Bmatrix} \tag{2.46}$$

The strong singular term added to the free term c equals the sum of the coefficients in one equation after changing their signs as equation (2.46) implies. The same can be done for the other directions y and z to have:

$$\mathbf{c}(P_i)\begin{Bmatrix}0\\1\\0\end{Bmatrix}+\sum_{e=1}^{E}\sum_{\substack{n=1\\g(n)=i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}0\\1\\0\end{Bmatrix}=-\sum_{e=1}^{E}\sum_{\substack{n=1\\g(n)\neq i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}0\\1\\0\end{Bmatrix}\;\&$$
$$\mathbf{c}(P_i)\begin{Bmatrix}0\\0\\1\end{Bmatrix}+\sum_{e=1}^{E}\sum_{\substack{n=1\\g(n)=i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}0\\0\\1\end{Bmatrix}=-\sum_{e=1}^{E}\sum_{\substack{n=1\\g(n)\neq i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}0\\0\\1\end{Bmatrix} \qquad (2.47)$$

An infinite medium is considered as a domain bounded by additional auxiliary surface S_R a sphere of radius R extends to infinity. Rigid body translation can be applied to this surface. An integral over this surface should be added to the left hand side of equation (2.47) as:

$$\mathbf{c}(\mathrm{P_i})\begin{Bmatrix}1\\0\\0\end{Bmatrix}+\sum_{e=1}^{E}\sum_{\substack{n=1\\g(n)=i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}1\\0\\0\end{Bmatrix}=-\sum_{e=1}^{E}\sum_{\substack{n=1\\g(n)\neq i}}^{N}\Delta\mathbf{T}_{ni}^{e}\begin{Bmatrix}1\\0\\0\end{Bmatrix}-\int_{S_R}\mathbf{T}(P_i,Q)dS \qquad (2.48)$$

This integral equals **(– I)** for the infinite medium and $(-\frac{1}{2}\mathbf{I})$ for the semi infinite medium. **I** is the unit matrix of dimension 3x3.

2.2.4.3 Numerical Integration for Two Dimensional Isoparametric Elements

The source of error in BEM is from either choosing a limited number of P=Q points at which Betti's theorem is satisfied, although theoretically it should be satisfied at every point on the boundary, or because of the numerical integration of kernel-shape functions products over the elements surfaces. The second type of errors is produced by numerical integration, increased because of the kernels' singularity. Gauss Quadrature numerical integration technique is used to obtain the coefficients $\Delta\mathbf{U}_{\mathrm{ni}}^{\mathrm{e}}$ and $\Delta\mathbf{T}_{\mathrm{ni}}^{\mathrm{e}}$ in equation (2.42). To reduce the error, Gauss integration points should be increased depending on the closeness of the load point P_i to the region of integration (the element surface), or in a more accurate expression, if P_i does not belong to the element, Gauss integration points are increased by the decrease of the ratio D_{min}/ L. D_{min} is the minimum distance between P_i and the element. L is the element length in the

integration direction (see Figure 2.11). Ebrwien, Duenser, and Moser [27] concluded that to integrate functions of the order o(1/r), number of Gauss integration points should not be less than 3 if the ratio $D_{min}/ L \leq 1.4025$ and should not be less than 4 Gauss integration points if the ratio $D_{min}/ L \leq$ **0.6736**. To integrate functions of the order $o(1/r^2)$ number of Gauss integration points should not be less than 3 if the ratio $D_{min}/ L \leq 2.3187$ and should not be less than 4 Gauss integration points if ratio $D_{min}/ L \leq$ **0.9709**. If ratio D_{min}/ L is less than the minimum limits in both cases (written in bold font) the region of integration should be divide into number of subdivisions until D_{min}/ L reach the minimum limit.

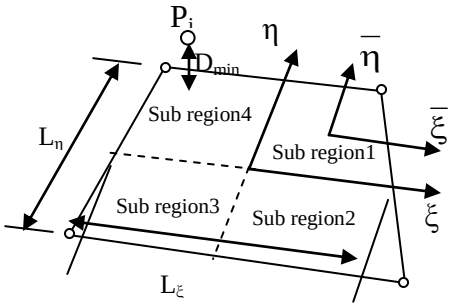


Figure2.11 Quadrilateral Element, Lengths', Minimum Distance, and Subdivisions

1-Coefficients $\Delta\mathbf{U}^e_{ni}$ and $\Delta\mathbf{T}^e_{ni}$ in the case where P_i is not a node in the 2D isoparametric element are computed by numerical integrations and according to equation (2.43) by:

$$\Delta\mathbf{U}^e_{ni} = \int_{-1}^{1}\int_{-1}^{1} N_n(\xi,\eta)\mathbf{U}(P_i, Q(\xi,\eta))J(\xi,\eta)d\xi d\eta$$
$$\Delta\mathbf{T}^e_{ni} = \int_{-1}^{1}\int_{-1}^{1} N_n(\xi,\eta)\mathbf{T}(P_i, Q(\xi,\eta))J(\xi,\eta)d\xi d\eta \qquad (2.49)$$

The above integrals using Gauss Quadrature become:

$$\Delta\mathbf{U}^e_{ni} \approx \sum_{m=1}^{M}\sum_{k=1}^{K} N_n(\xi_m,\eta_k)\mathbf{U}(P_i, Q(\xi_m,\eta_k))J(\xi_m,\eta_k)w_m w_k$$
$$\Delta\mathbf{T}^e_{ni} \approx \sum_{m=1}^{M}\sum_{k=1}^{K} N_n(\xi_m,\eta_k)\mathbf{T}(P_i, Q(\xi_m,\eta_k))J(\xi_m,\eta_k)w_m w_k \qquad (2.50)$$

M and K are the number of Gauss integration points in ξ and η directions respectively depending on D_{min}/ L_ξ and D_{min}/ L_η ratios (see Figure 2.11). ξ_m, η_k, w_m, w_k are Gauss integration points coordinates and weights in ξ and η directions respectively. J is the jacobian of transformation between global or Cartesian and intrinsic coordinates.

If P_i is very close to the element that D_{min}/ L_ξ or D_{min}/ L_η are less than $(D_{min}/ L)_{min,}$ region of integration should be divided into sub regions. Numbers of sub regions are given by:

$$\begin{aligned} N_\xi &= Int[(D_{\min}/L)_{\min}/(D_{\min}/L_\xi)] \\ N_\eta &= Int[(D_{\min}/L)_{\min}/(D_{\min}/L_\eta)] \end{aligned} \tag{2.51}$$

Equations (2.50) become:

$$\begin{aligned} \Delta\mathbf{U}^e_{ni} &\approx \sum_{l=1}^{N_\xi}\sum_{j=1}^{N_\eta}\sum_{m=1}^{M(l)}\sum_{k=1}^{K(j)} N_n(\bar{\xi}_m,\bar{\eta}_k)\mathbf{U}(P_i, Q(\bar{\xi}_m,\bar{\eta}_k))J(\bar{\xi}_m,\bar{\eta}_k)\bar{J}.w_m w_k \\ \Delta\mathbf{T}^e_{ni} &\approx \sum_{l=1}^{N_\xi}\sum_{j=1}^{N_\eta}\sum_{m=1}^{M(l)}\sum_{k=1}^{K(j)} N_n(\bar{\xi}_m,\bar{\eta}_k)\mathbf{T}(P_i, Q(\bar{\xi}_m,\bar{\eta}_k))J(\bar{\xi}_m,\bar{\eta}_k)\bar{J}.w_m w_k \end{aligned} \tag{2.52}$$

M(l) and K(j) are number of integration points in ξ and η directions for the sub region.

ξ, η are given in term of sub regional intrinsic coordinates $\bar{\xi}, \bar{\eta}$ by:

$$\xi = \frac{1}{2}(\xi_1+\xi_2) + \frac{\bar{\xi}}{N_\xi} \;\&\; \eta = \frac{1}{2}(\eta_1+\eta_2) + \frac{\bar{\eta}}{N_\eta} \tag{2.53}$$

$\xi_1, \xi_2 \text{and} \eta_1, \eta_2$ are elemental intrinsic coordinates that defines the extremes of the sub regions.

The jacobian is given by:

$$\left|\bar{J}\right| = \frac{\partial\xi}{\partial\bar{\xi}}\frac{\partial\eta}{\partial\bar{\eta}} = \frac{1}{N_\xi . N_\eta} \tag{2.54}$$

It has been discussed above how to do numerical integration if P_i is not a node in the element (the region of integration).

2-If P_i is a node of the element a two cases will be differentiated: The first case when we have P_i =g(n), the load point coincides with the field point, in this case only $\Delta\mathbf{U}^e_{ni}$ can be computed numerically but after dividing the element into triangular sub elements. $\Delta\mathbf{T}^e_{ni}$ Can be computed

only in the sense of the Cauchy principal value and can be obtained when added to the free term using equation (2.47) or equation (2.48). The second case if $P_i \neq g(n)$ but P_i still belongs to the element in this case. In this case both coefficients $\Delta\mathbf{U}^e_{ni}$ and $\Delta\mathbf{T}^e_{ni}$ can be determined by dividing the element into triangular sub elements. In each sub element local intrinsic coordinates $\bar{\xi}, \bar{\eta}$ are used by which the jacobian of transformation $\bar{J}$ from elemental intrinsic coordinates ξ, η to $\bar{\xi}, \bar{\eta}$ approaches zero at node P_i. Number of sub elements is two if P_i is at a corner node of the element and three if P_i is at a mid side node of the element as shown in Figure 2.12.

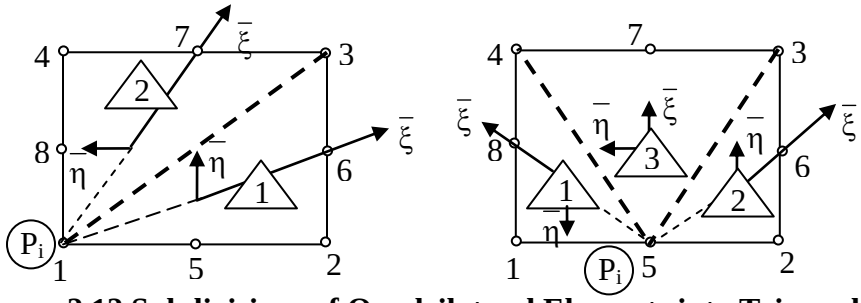


Figure 2.12 Subdivisions of Quadrilateral Elements into Triangular Sub-elements. Pi at Corner Node, Node 1 as an Example (left). Pi at Mid-side Node, Node 5 as an Example (right)

Coefficients $\Delta\mathbf{U}^e_{ni}$ and $\Delta\mathbf{T}^e_{ni}$ are given according to the method above obtained by:

$$\Delta\mathbf{U}^e_{ni} \approx \sum_{s=1}^{2(3)}\sum_{m=1}^{M}\sum_{k=1}^{K} N_n(\bar{\xi}_m, \bar{\eta}_k)\mathbf{U}(P_i, Q(\bar{\xi}_m, \bar{\eta}_k))J(\bar{\xi}_m, \bar{\eta}_k)\bar{J}(\bar{\xi}_m, \bar{\eta}_k)w_m w_k$$
$$\Delta\mathbf{T}^e_{ni} \approx \sum_{s=1}^{2(3)}\sum_{m=1}^{M}\sum_{k=1}^{K} N_n(\bar{\xi}_m, \bar{\eta}_k)\mathbf{T}(P_i, Q(\bar{\xi}_m, \bar{\eta}_k))J(\bar{\xi}_m, \bar{\eta}_k)\bar{J}(\bar{\xi}_m, \bar{\eta}_k)w_m w_k \quad (2.55)$$
$$g(n)\neq P_i$$

Relationships between $\bar{\xi}, \bar{\eta}$ and ξ, η are given by:

$$\xi = \sum_{n=1}^{3} \bar{N}_n(\bar{\xi}, \bar{\eta})\xi_{l(n)}$$
$$\eta = \sum_{n=1}^{3} \bar{N}_n(\bar{\xi}, \bar{\eta})\eta_{l(n)} \quad (2.56)$$

Where l(n) node is number of the n^{th} triangular sub element node. Sub element shape functions are given by:

$$\begin{aligned}\bar{N}_1 &= \frac{1}{4}(1+\bar{\xi})(1-\bar{\eta}) \\ \bar{N}_2 &= \frac{1}{4}(1+\bar{\xi})(1+\bar{\eta}) \\ \bar{N}_3 &= \frac{1}{4}(1-\bar{\xi})\end{aligned} \tag{2.57}$$

The jacobian of transformation $\bar{J}$ is given by:

$$\bar{J} = \frac{\partial\xi}{\partial\bar{\xi}}\frac{\partial\eta}{\partial\bar{\eta}} - \frac{\partial\eta}{\partial\bar{\xi}}\frac{\partial\xi}{\partial\bar{\eta}} \tag{2.58}$$

2.2.5 Assembly of System of Equations

The final step is to build a system of equations each is similar to equation (2.42):

$$\mathbf{cu}(P_i) + \sum_{e=1}^{E}\sum_{n=1}^{N}\Delta\mathbf{T}_{ni}^{e}\mathbf{u}_n^e = \sum_{e=1}^{E}\sum_{n=1}^{N}\Delta\mathbf{U}_{ni}^{e}\mathbf{t}_n^e \tag{2.42}$$

The total number of equations for 3D elasticity problems is 3I, where (I) is the total number of nodes or collocation points P_i.

For programming necessities let's rewrite the equations as a sum of elements' coefficients; in form of matrices as the following:

$$[\Delta T]\{u\} = [\Delta U]\{t\} \tag{2.59}$$

$\{u\}$ and $\{t\}$ are global nodal displacement and traction vectors. The nodes are numbered for these vectors globally. $[\Delta T]$ and $[\Delta U]$ are global coefficient matrices of dimensions 3Ix3I built by placing elements' coefficients $\Delta\mathbf{U}_{ni}^{e}$ and $\Delta\mathbf{T}_{ni}^{e}$ components into the global matrices according to connectivity or incidences of the element and according to boundary conditions. $[\Delta T]$ or $[\Delta U]$ are non symmetric fully populated coefficient matrices.

If the prescribed boundary values are substituted in equation (2.59) it will take the form:

$$[\Delta T]\{u\} = \{F\} \tag{2.60}$$

Unknown boundary values are computed by solving equation (2.60).

2.2.6 Computation of Displacement and Stress Values inside the Domain

Equation (2.32) can be used to compute the displacement vector at point P_a inside the domain. It is easily implemented by having both boundary values, displacement and traction vectors, at the boundary nodes as the following:

$$\mathbf{u}(P_a) = \int_S \mathbf{U}(P_a, Q)\mathbf{t}(Q)dS - \int_S \mathbf{T}(P_a, Q)\mathbf{u}(Q)dS \tag{2.61}$$

the integrals in equations (2.61) are evaluated numerically in the following discretized form:

$$\mathbf{u}(P_a) = \sum_{e=1}^{E}\sum_{n=1}^{N} \Delta\mathbf{U}_{na}^{e}\mathbf{t}_{n}^{e} - \sum_{e=1}^{E}\sum_{n=1}^{N} \Delta\mathbf{T}_{na}^{e}\mathbf{u}_{n}^{e} \tag{2.62}$$

Coefficients $\Delta\mathbf{U}_{na}^{e}$ and $\Delta\mathbf{T}_{na}^{e}$ using Gauss Quadrature are given by:

$$\begin{aligned} \Delta\mathbf{U}_{na}^{e} &\approx \sum_{m=1}^{M}\sum_{k=1}^{K} N_n(\xi_m, \eta_k)\mathbf{U}(P_a, Q(\xi_m, \eta_k))J(\xi_m, \eta_k)w_m w_k \\ \Delta\mathbf{T}_{na}^{e} &\approx \sum_{m=1}^{M}\sum_{k=1}^{K} N_n(\xi_m, \eta_k)\mathbf{T}(P_a, Q(\xi_m, \eta_k))J(\xi_m, \eta_k)w_m w_k \end{aligned} \tag{2.63}$$

To have the stress at internal points equation (2.61) should be derived first to have the strain tensor. Because it is easier to understand and derive let us write equation (2.1) in the following form:

$$\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{u} \tag{2.64}$$

Where **B** is a differential operator given by:

$$\mathbf{B} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \tag{2.65}$$

The resulted matrix in equation (2.64) is a pseudo-strain vector of 6 components, which includes three normal strain components ($\varepsilon_x, \varepsilon_y \text{and} \varepsilon_z$) and three shear strain components ($\gamma_{xy} = \gamma_{yx}, \gamma_{yz} = \gamma_{zy} \text{and} \gamma_{zx} = \gamma_{xz}$). Equation (2.61) after derivation becomes:

$$\boldsymbol{\varepsilon} = \mathbf{Bu}(P_a) = \int_S \mathbf{BU}(P_a, Q)\mathbf{t}(Q)dS - \int_S \mathbf{BT}(P_a, Q)\mathbf{u}(Q)dS \tag{2.66}$$

If equation (2.66) is multiplied by **D** tensor, **D** is the constitutive tensor of dimensions 6x6 that relates the stress with strain vectors for isotropic linear elastic materials ($\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}$), a pseudo-stress vector ($\sigma_x, \sigma_y, \sigma_z, \sigma_{xy} = \sigma_{yx}, \sigma_{yz} = \sigma_{zy} \text{and} \sigma_{zx} = \sigma_{xz}$) is resulted as the following:

$$\boldsymbol{\sigma} = \int_S \mathbf{S}(P_a, Q)\mathbf{t}(Q)dS - \int_S \mathbf{R}(P_a, Q)\mathbf{u}(Q)dS \tag{2.67}$$

Where **S** = **DBU** (P_a, Q) and **R** = **DBT** (P_a, Q). Both matrices are of dimensions 6x3 and given by:

$$S_{ijr} = \frac{1}{8\pi(1-\nu)r^2}\left[3r_{,i}r_{,j}r_{,r} + (1-2\nu)(\delta_{ri}r_{,j} + \delta_{rj}r_{,i} - \delta_{ij}r_{,r})\right] \tag{2.68}$$

$$R_{ijr} = \frac{\mu}{4\pi(1-\nu)r^3}\begin{bmatrix} 3r_{,s}n_s[(1-2\nu)r_{,r}\delta_{ij} + \nu(r_{,i}\delta_{rj} + r_{,j}\delta_{ri}) - 5r_{,r}r_{,i}r_{,j}] + 3\nu[r_{,j}r_{,r}n_i + r_{,i}r_{,r}n_j] \\ + (1-2\nu)[3r_{,i}r_{,j}n_r + \delta_{ri}n_j + \delta_{rj}n_i] - (1-4\nu)\delta_{ij}n_r \end{bmatrix}$$

Special care should be taken when the load point approaches the boundary. If P_i is very close to the element that D_{min}/ L is less than $(D_{min}/ L)_{min}$ the element should be divided into sub regions in similar procedure explained before.

2.2.7 The Stress at the Boundary

Although three stress components at the boundary are already obtained by having the traction vectors at boundary nodes, the other three stress components are still needed to be obtained. The following stress components using the Cauchy rule ($\bar{t}_i = \bar{\sigma}_{ji}\bar{n}_j$) are given by:

$$t_{\bar{x}} = \tau_{\bar{x}\bar{z}}; t_{\bar{y}} = \tau_{\bar{y}\bar{z}}; t_{\bar{z}} = \sigma_{\bar{z}} \tag{2.69}$$

Where $\bar{x}, \bar{y}, \bar{z}$ are local coordinates of a boundary element,

The other three stress components are defined using equation (2.2) as:

$$\begin{aligned} \sigma_{\bar{x}} &= \frac{2\mu}{(1-\nu)}(\varepsilon_{\bar{x}} + \nu\varepsilon_{\bar{y}}) + \frac{\nu}{(1-\nu)} t_{\bar{z}} \\ \sigma_{\bar{y}} &= \frac{2\mu}{(1-\nu)}(\varepsilon_{\bar{y}} + \nu\varepsilon_{\bar{x}}) + \frac{\nu}{(1-\nu)} t_{\bar{z}} \\ \tau_{\bar{x}\bar{y}} &= 2\mu\varepsilon_{\bar{x}\bar{y}} \end{aligned} \tag{2.70}$$

Strain components $\varepsilon_{\bar{x}}, \varepsilon_{\bar{y}}$ and $\varepsilon_{\bar{x}\bar{y}}$ can be obtained using equation (2.1), but after projecting the displacement vector **u** prescribed or computed in global coordinates on the local coordinate system.

2.2.8 Boundary Element Method in the Semi-Infinite Domain

Problems in isotropic linear elastic semi-infinite medium can be solved either by using: Melan's fundamental solution in half plane medium for 2D plain strain or plain stress problems, Mindlin's solution in semi-infinite half space medium for 3D problems or Kelvin's fundamental solution in infinite medium and subdivide the ground surface into finite and infinite boundary elements.

2.2.8.1 BEM in Semi-Infinite Medium Using Mindlin's Solution

Mindlin obtained the fundamental solutions [4] for a concentrated force applied in the three directions x, y and z in half space medium with free traction surface $\bar{S}$ (see Figure 2.13) by superposition of eighteen nuclei of strain in infinite medium, six nuclei of strain for each force direction (three groups), all derived from Kelvin's solution. Kelvin's force applied at (0, 0, +c) was the first solution in each group of the nuclei of strain [5]. The rest of the nuclei of strain such as double force, center of compression, a line of center of compressions and others all were applied at (0, 0, -c) the image of Kelvin's force application point with respect to the surface $\bar{S}$. These nuclei of strain are incorporated in the solution to satisfy the free traction surface condition [5].

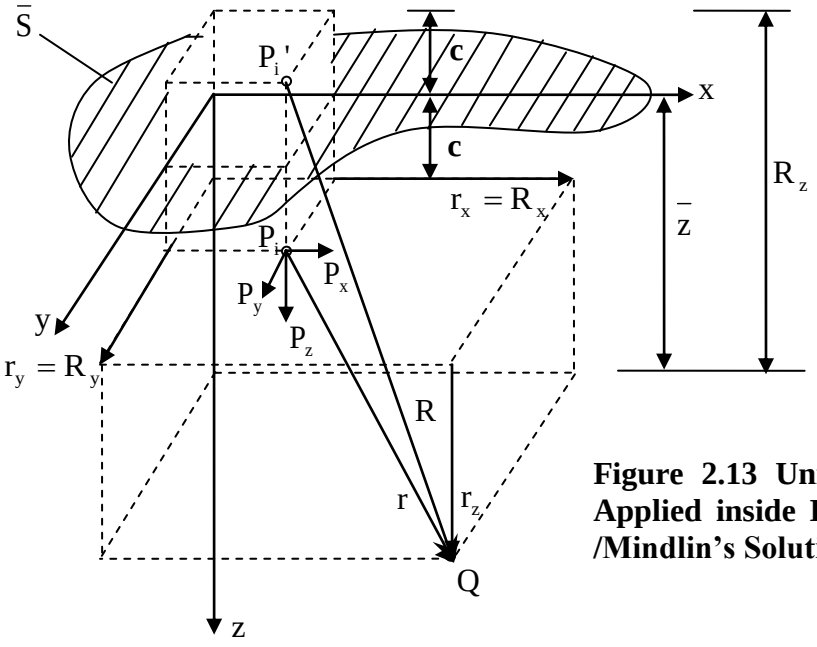


Figure 2.13 Unit Point Load Applied inside Half Space [5] /Mindlin's Solution/.

Mindlin's solutions of displacements and stresses everywhere in the semi-infinite medium are written, for the abovementioned reasons, as the sum of two parts, the first is Kelvin's solution and the second is a complementary solution. The complementary solutions are [5]:

$$U^c_{xx} = K_d\left[\frac{1}{R} + \frac{(3-4\nu)r_x^2}{R^3} + \frac{2c\bar{z}}{R^3}\left(1-\frac{3r_x^2}{R^2}\right) + \frac{4(1-\nu)(1-2\nu)}{R+R_z}\left(1-\frac{r_x^2}{R(R+R_z)}\right)\right]$$

$$U^c_{xy} = K_d r_x r_y\left[\frac{3-4\nu}{R^3} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)^2} - \frac{6c\bar{z}}{R^5}\right]$$

$$U^c_{xz} = K_d r_x\left[\frac{(3-4\nu)r_z}{R^3} + \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)} - \frac{6c\bar{z}R_z}{R^5}\right]$$

$$U^c_{yx} = U^c_{xy}, U^c_{yz} = \frac{r_y}{r_x}U^c_{xz}$$

$$U^c_{yy} = K_d\left[\frac{1}{R} + \frac{(3-4\nu)r_y^2}{R^3} + \frac{2c\bar{z}}{R^3}\left(1-\frac{3r_y^2}{R^2}\right) + \frac{4(1-\nu)(1-2\nu)}{R+R_z}\left(1-\frac{r_y^2}{R(R+R_z)}\right)\right]$$

$$U^c_{zx} = K_d r_x\left[\frac{(3-4\nu)r_z}{R^3} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)} + \frac{6c\bar{z}R_z}{R^5}\right]$$

$$U^c_{zy} = \frac{r_y}{r_x}U^c_{zx}$$

$$U^c_{zz} = K_d\left[\frac{8(1-\nu)^2-(3-4\nu)}{R} + \frac{(3-4\nu)R_z^2 - 2c\bar{z}}{R^3} + \frac{6c\bar{z}R_z^2}{R^5}\right] \qquad (2.71.a)$$

$$K_d = \frac{1}{16\pi(1-\nu)G}$$

$$\sigma^{c}_{xxx} = K_s r_x \left[\frac{(1-2\nu)(5-4\nu)}{R^3} - \frac{3(3-4\nu)r_x^2}{R^5} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)^2}\left(3 - \frac{r_x^2(3R+R_z)}{R^2(R+R_z)}\right) + \frac{6c}{R^5}\left(3c - (3-2\nu)R_z + \frac{5r_x^2\bar{z}}{R^2}\right)\right]$$

$$\sigma^{c}_{xyx} = K_s r_y \left[\frac{(1-2\nu)}{R^3} - \frac{3(3-4\nu)r_x^2}{R^5} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)^2}\left(1 - \frac{r_x^2(3R+R_z)}{R^2(R+R_z)}\right) - \frac{6c\bar{z}}{R^5}\left(1 - \frac{5r_x^2}{R^2}\right)\right]$$

$$\sigma^{c}_{yyx} = K_s r_x \left[\frac{(1-2\nu)(3-4\nu)}{R^3} - \frac{3(3-4\nu)r_y^2}{R^5} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)^2}\left(1 - \frac{r_y^2(3R+R_z)}{R^2(R+R_z)}\right) + \frac{6c}{R^5}\left(c - (1-2\nu)R_z + \frac{5r_y^2\bar{z}}{R^2}\right)\right]$$

$$\sigma^{c}_{zxx} = K_s \left[\frac{(1-2\nu)r_z}{R^3} - \frac{3(3-4\nu)r_x^2 R_z}{R^5} - \frac{6c}{R^5}\left(\bar{z}R_z - (1-2\nu)r_x^2 - \frac{5r_x^2\bar{z}R_z}{R^2}\right)\right]$$

$$\sigma^{c}_{zyx} = K_s r_x r_y \left[-\frac{3(3-4\nu)R_z}{R^5} + \frac{6c}{R^5}\left(1 - 2\nu + \frac{5\bar{z}R_z}{R^2}\right)\right] \qquad (2.71.b)$$

$$\sigma^{c}_{zzx} = K_s r_x \left[-\frac{(1-2\nu)}{R^3} - \frac{3(3-4\nu)R_z^2}{R^5} + \frac{6c}{R^5}\left(c + (1-2\nu)R_z + \frac{5\bar{z}R_z^2}{R^2}\right)\right]$$

$$\sigma^c_{xxy} = K_s r_y \left[\frac{(1-2\nu)(3-4\nu)}{R^3} - \frac{3(3-4\nu)r_x^2}{R^5} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)^2}\left(1 - \frac{r_x^2(3R+R_z)}{R^2(R+R_z)}\right) + \frac{6c}{R^5}\left(c - (1-2\nu)R_z + \frac{5r_x^2\bar{z}}{R^2}\right)\right]$$

$$\sigma^c_{xyy} = K_s r_x \left[\frac{(1-2\nu)}{R^3} - \frac{3(3-4\nu)r_y^2}{R^5} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)^2}\left(1 - \frac{r_y^2(3R+R_z)}{R^2(R+R_z)}\right) - \frac{6c\bar{z}}{R^5}\left(1 - \frac{5r_y^2}{R^2}\right)\right]$$

$$\sigma^c_{yyy} = K_s r_y \left[\frac{(1-2\nu)(5-4\nu)}{R^3} - \frac{3(3-4\nu)r_y^2}{R^5} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)^2}\left(3 - \frac{r_y^2(3R+R_z)}{R^2(R+R_z)}\right) + \frac{6c}{R^5}\left(3c - (3-2\nu)R_z + \frac{5r_y^2\bar{z}}{R^2}\right)\right]$$

$$\sigma^c_{zyy} = K_s \left[\frac{(1-2\nu)r_z}{R^3} - \frac{3(3-4\nu)r_y^2 R_z}{R^5} - \frac{6c}{R^5}\left(\bar{z}R_z - (1-2\nu)r_y^2 - \frac{5r_y^2\bar{z}R_z}{R^2}\right)\right]$$

$$\sigma^c_{zzy} = \frac{r_y}{r_x}\sigma^c_{zzx}, \sigma^c_{zxy} = \sigma^c_{zyx}. \text{where}: K_s = \frac{1}{8\pi(1-\nu)} \text{and} T^c_{ki} = \sigma^c_{ijk} n_j$$

$$\sigma^c_{xxz} = K_s \left[\frac{(1-2\nu)(3r_z - 4\nu R_z)}{R^3} - \frac{3(3-4\nu)r_x^2 r_z - 6cR_z[(1-2\nu)\bar{z} - 2\nu c]}{R^5} - \frac{30cr_x^2\bar{z}R_z}{R^7} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)}\left(1 - \frac{r_x^2}{R(R+R_z)} - \frac{r_x^2}{R^2}\right)\right]$$

$$\sigma^c_{xyz} = K_s r_x r_y \left[-\frac{3(3-4\nu)r_z}{R^5} + \frac{4(1-\nu)(1-2\nu)}{R^2(R+R_z)}\left(\frac{1}{R+R_z} + \frac{1}{R}\right) - \frac{30c\bar{z}R_z}{R^7}\right]$$

$$\sigma^c_{yyz} = K_s \left[\frac{(1-2\nu)(3r_z - 4\nu R_z)}{R^3} - \frac{3(3-4\nu)r_y^2 r_z - 6cR_z[(1-2\nu)\bar{z} - 2\nu c]}{R^5} - \frac{30cr_y^2\bar{z}R_z}{R^7} - \frac{4(1-\nu)(1-2\nu)}{R(R+R_z)}\left(1 - \frac{r_y^2}{R(R+R_z)} - \frac{r_y^2}{R^2}\right)\right]$$

$$\sigma^c_{zxz} = K_s r_x \left[\frac{(1-2\nu)}{R^3} - \frac{3(3-4\nu)\bar{z}R_z - 3c(3\bar{z}+c)}{R^5} - \frac{30c\bar{z}R_z^2}{R^7}\right]$$

$$\sigma^c_{zyz} = K_s r_y \left[\frac{(1-2\nu)}{R^3} - \frac{3(3-4\nu)\bar{z}R_z - 3c(3\bar{z}+c)}{R^5} - \frac{30c\bar{z}R_z^2}{R^7}\right]$$

$$\sigma^c_{zzz} = K_s \left[\frac{(1-2\nu)r_z}{R^3} - \frac{3(3-4\nu)\bar{z}R_z^2 - 3cR_z(5\bar{z}-c)}{R^5} - \frac{30c\bar{z}R_z^3}{R^7}\right] \qquad (2.71.c)$$

Boundary Integral Equations

BIE for semi-infinite problems is not different from the equation of Kelvin's region:

$$\mathbf{cu}(P) = \int_S \mathbf{U}(P,Q)\mathbf{t}(Q)dS - \int_S \mathbf{T}(P,Q)\mathbf{u}(Q)dS$$

T and **U** are Mindlin's fundamental solutions. The second integral is may be modified for two cases discussed below, we often see in half space problems, as the following:

$$\mathbf{cu}(P) = \int_S \mathbf{U}(P,Q)\mathbf{t}(Q)dS - \int_{S'} \mathbf{T}(P,Q)\mathbf{u}(Q)dS \qquad (2.72)$$

The first case is that a cavity or a domain has a boundary surface S, part of it S-S` coincide with the free traction surface $\bar{S}$ (see Figure 2.14-a). Integral of Mindlin's fundamental solution **T** over the boundary S-S` is zero due to the surface free traction condition satisfied by the solution. If S-S` is loaded and when the collocation points $P_i$ are located at that surface (c=0), weak singular Mindlin's fundamental solution **U** can be integrated numerically as explained before with no further difficulty. If S-S` is not loaded, (**t**=0) this weak singularity will vanish too. Points at this boundary are considered as internal points [5].

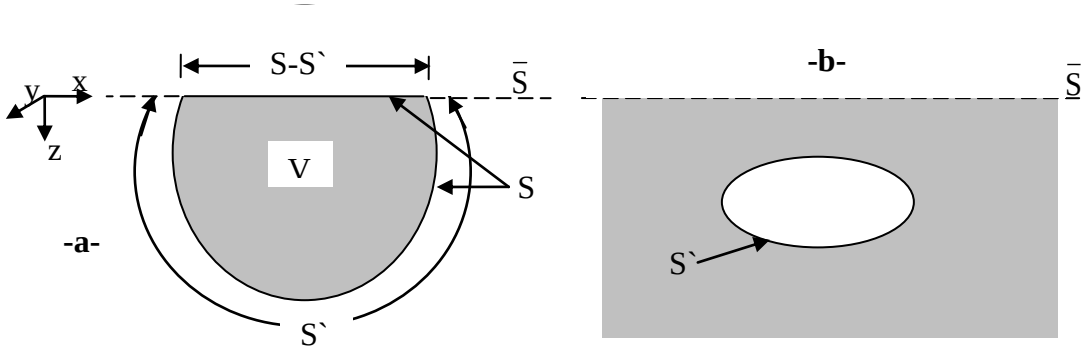


Figure 2.14 Semi-Infinite Domains

The second case is if the domain boundary does not coincide with the surface $\bar{S}$ (see Figure 2.14-b). If the surface $\bar{S}$ is partially loaded on surface S-S`, the same discussion as above still applies here, and if surface $\bar{S}$ is totally unloaded, boundary S equals S` in this case. Singularities in Mindlin's fundamental solution are because of Kelvin's part, not because of the

complementary part (see equations 2.71). For that reason integrations in the sense of the Cauchy principal value for Kelvin's solution can be applied to Mindlin's solution and equations (2.36) and (2.37) can be used too. Numerical integration procedures explained before are the same in both cases except that in equation (2.48) the integral $\int_{S_R} \mathbf{T}(P_i, Q)dS$ over the semi sphere surface S_R with radius extends to infinity for Mindlin's solution **T** equals **–I .**

2.2.8.2 BEM in Semi-Infinite Medium Using Infinite Boundary Elements

Although the first method using Mindlin's fundamental solution is preferred for semi-infinite problems for simple geometry, due to its higher accuracy, it is still needed for complex geometry and multi-region problems to discretize the interfaces (as boundaries extend to infinity) into isoparametric finite and infinite boundary elements. The second method, as suggested by W. Moser et al. [6] , uses Kelvin's fundamental solution in a semi-infinite medium by subdividing the ground surface into isoparametric finite BEs and infinite BEs. Because of the different nature of finite and infinite BEs as will be explained, the discretized BIE in equation (2.42) can be rewritten as [6]:

$$\mathbf{cu}(P_i) + \sum_{e=1}^{finiteBEs}\sum_{n=1}^{N}\Delta\mathbf{T}_{ni}^{e}\mathbf{u}_{n}^{e} + \sum_{e=1}^{infiniteBEs}\sum_{n=1}^{Np}{}^{\infty}\Delta\mathbf{T}_{ni}^{e}\mathbf{u}_{n}^{e} = \sum_{e=1}^{finiteBEs}\sum_{n=1}^{N}\Delta\mathbf{U}_{ni}^{e}\mathbf{t}_{n}^{e} + \sum_{e=1}^{infiniteBEs}\sum_{n=1}^{Np}{}^{\infty}\Delta\mathbf{U}_{ni}^{e}\mathbf{t}_{n}^{e} \qquad (2.73)$$

Where $\Delta\mathbf{U}_{ni}^{e}$ and $\Delta\mathbf{T}_{ni}^{e}$ are given by equations (2.43) and ${}^{\infty}\Delta\mathbf{U}_{ni}^{e}$ and ${}^{\infty}\Delta\mathbf{T}_{ni}^{e}$ are given by:

$$\begin{aligned} {}^{\infty}\Delta\mathbf{U}_{ni}^{e} &= \int_{S_e} {}^{\infty}N_n^t(\xi,\eta)\mathbf{U}(P_i,\xi,\eta)dS(\xi,\eta) \\ {}^{\infty}\Delta\mathbf{T}_{ni}^{e} &= \int_{S_e} {}^{\infty}N_n^u(\xi,\eta)\mathbf{T}(P_i,\xi,\eta)dS(\xi,\eta) \end{aligned} \qquad (2.74)$$

Unlike isoparametric finite BEs, infinite BEs shape or mapping functions ${}^{\infty}N_n$ are different from interpolation functions ${}^{\infty}N_n^u$ and ${}^{\infty}N_n^t$. Shape functions ${}^{\infty}N_n$ map the finite surface using intrinsic coordinate η and ξ that varies between 1 and -1, into infinite surface (see Figure

2.15). The number of nodes N used for mapping is different from the number of nodes Np used for displacement and traction interpolation. Decay or asymptotic behaviour in infinite elements is achieved by functions ${}^{\infty}N_n^u$ and ${}^{\infty}N_n^t$ which interpolate the displacement and traction respectively. Interpolation functions make displacement and traction vanish at infinity in η direction [6].

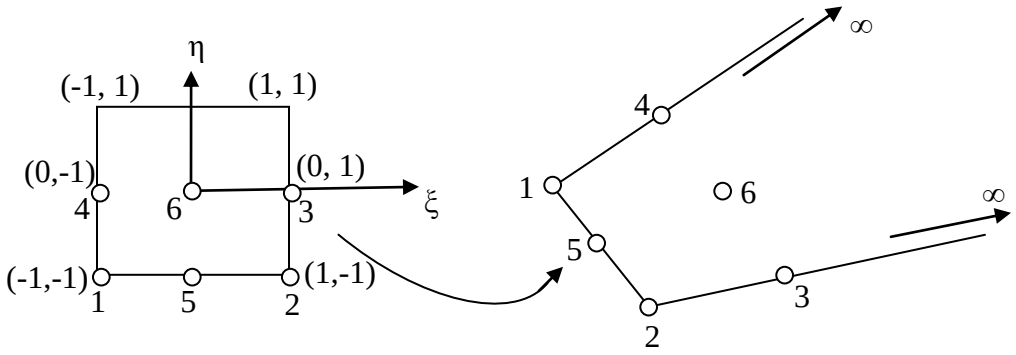


Figure 2.15 Two Dimensional Infinite Element [6]

Two kinds of infinite BEs can be used. The first is four-node elements with linear interpolation in ξ direction. Shape and interpolation functions for linear elements are given by [6]:

$$
\begin{aligned}
&{}^{\infty}N_1 = \frac{1}{2}(1-\xi)\,{}^{\infty}N_1^{(1-D)}, {}^{\infty}N_2 = \frac{1}{2}(1+\xi)\,{}^{\infty}N_1^{(1-D)} \\
&{}^{\infty}N_3 = \frac{1}{2}(1+\xi)\,{}^{\infty}N_3^{(1-D)}, {}^{\infty}N_4 = \frac{1}{2}(1+\xi)\,{}^{\infty}N_3^{(1-D)} \\
&{}^{\infty}N_1^u = \frac{1}{2}(1-\eta).[\frac{1}{2}(1-\xi)], {}^{\infty}N_2^u = \frac{1}{2}(1-\eta).[\frac{1}{2}(1+\xi)] \\
&{}^{\infty}N_1^t = \frac{1}{4}(1-\eta)^2.[\frac{1}{2}(1-\xi)], {}^{\infty}N_2^t = \frac{1}{4}(1-\eta)^2.[\frac{1}{2}(1+\xi)]
\end{aligned}
\tag{2.75}
$$

${}^{\infty}N_1^{(1-D)}, {}^{\infty}N_1^{(1-D)}$ are one-dimensional mapping functions given by [6]:

$$
{}^{\infty}N_1^{(1-D)} = -\frac{2\eta}{1-\eta}, {}^{\infty}N_3^{(1-D)} = \frac{1+\eta}{1-\eta} \tag{2.76}
$$

The second kind of infinite BEs is six-node elements with quadratic interpolation in ξ direction. Shape and interpolation functions for quadratic elements are given by [6]:

$$
\begin{aligned}
&{}^{\infty}N_1 = \frac{\xi}{2}(\xi-1)\,{}^{\infty}N_1^{(1-D)}, {}^{\infty}N_2 = \frac{\xi}{2}(\xi+1)\,{}^{\infty}N_1^{(1-D)} \\
&{}^{\infty}N_3 = \frac{\xi}{2}(\xi+1)\,{}^{\infty}N_3^{(1-D)}, {}^{\infty}N_4 = \frac{\xi}{2}(\xi-1)\,{}^{\infty}N_3^{(1-D)} \\
&{}^{\infty}N_5 = (1-\xi^2)\,{}^{\infty}N_1^{(1-D)}, {}^{\infty}N_6 = (1-\xi^2)\,{}^{\infty}N_3^{(1-D)} \\
&{}^{\infty}N_1^{u} = \frac{1}{2}(1-\eta).[\frac{1}{2}\xi(\xi-1)], {}^{\infty}N_2^{u} = \frac{1}{2}(1-\eta).[\frac{1}{2}\xi(\xi+1)], {}^{\infty}N_5^{u} = \frac{1}{2}(1-\eta).[(1-\xi^2)] \\
&{}^{\infty}N_1^{t} = \frac{1}{4}(1-\eta)^2.[\frac{1}{2}\xi(\xi-1)], {}^{\infty}N_2^{t} = \frac{1}{4}(1-\eta)^2.[\frac{1}{2}\xi(\xi+1)], {}^{\infty}N_5^{t} = \frac{1}{4}(1-\eta)^2.[(1-\xi^2)]
\end{aligned}
\tag{2.77}
$$

Strongly singular term of traction kernel when load point P_i belongs to the element as region of integration and when i=g(n) can be computed using a direct method developed by Guiggiani and Gigante [7]. Cauchy-principal-value integral is given by Pereira & Parreira [8] as:

$$I_{nij} = \oint_{S_n} T_{ij}(P_i, Q)\,{}^{\infty}N_u^n dS(Q) \tag{2.77}$$

S_n is the region of integration that contains P_i.

Intrinsic coordinates η and ξ are transformed into polar coordinates ρ and θ (see Figure 2.16) given by [8]:

$$\xi_\gamma = \xi_\beta + \rho\cos(\theta), \eta_\gamma = \eta_\beta + \rho\sin(\theta) \tag{2.78}$$

Where β (ξ_β, η_β) and γ (ξ_γ, η_γ) are the images of load and field points P and Q respectively in the mapped element. Intrinsic domain should be subdivided into 2 or 3 triangle sub elements according to the source point position (see Figure12, and section 2. 2.4.3).

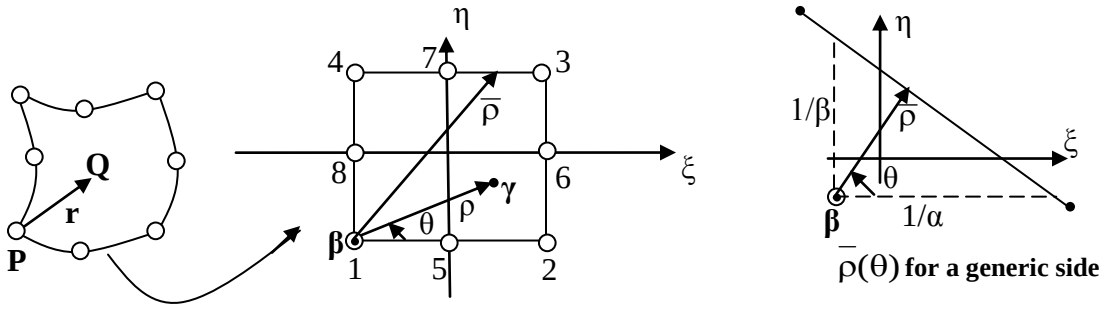


Figure 2.16 Coordinate Transformation into Polar Coordinates ρ and θ [8]

Integral in equation (2.77) is computed numerically by [7]:

$$I_{nij} = \sum_{s=1}^{2(3)\text{triangles}} \left\{ \int_{\theta 1}^{\theta 2} \int_{0}^{\bar{\rho}(\theta)} \left[F_{nij}(\rho,\theta) - \frac{f_{nij}(\theta)}{\rho} \right] d\rho d\theta + \int_{\theta 1}^{\theta 2} f_{nij}(\theta) \ln\left[\bar{\rho}(\theta) A(\theta)\right] d\theta \right\} \quad (2.79)$$

Integration limits should be transformed into Gauss quadrature limits 1 and -1 according to:

$$\rho = \frac{\bar{\rho}(\theta)}{2}(s+1), \theta = \frac{\theta_2 - \theta_1}{2} t + \frac{\theta_2 + \theta_1}{2} \quad \text{and} \quad J_1 = \frac{\bar{\rho}(\theta)}{2} \frac{\theta_2 - \theta_1}{2} \quad (2.80.a)$$

Where $-1 \le s \le +1 and -1 \le t \le +1$

Angles $\theta_2; \theta_1$ values depend on the triangle number (see Figure 2.12) and on source point position. J_1 is the Jacobian of transformation from ρ & θ to s & t.

Equations 2.80 to 2.82 as presented by Pereira & Parreira [8] are:

$$F_{nij} = T_{ij}\left[P_i, \gamma(\rho,\theta)\right] N_n^u\left[\gamma(\rho,\theta)\right] J\left[\gamma(\rho,\theta)\right] \rho \quad (2.80.b)$$

$$A(\theta) = \sqrt{\sum_{1}^{3} \left[A_i(\theta)\right]^2}; A_i(\theta) = \left(\frac{\partial x_i}{\partial \xi}\right)_{\beta=\gamma} \cos\theta + \left(\frac{\partial x_i}{\partial \eta}\right)_{\beta=\gamma} \sin\theta \quad (2.80.c)$$

$\bar{\rho}(\theta)$ For generic side (see Figure 2.16):

$$\bar{\rho}(\theta) = \frac{1}{\alpha\cos\theta + \beta\sin\theta} \quad (2.81)$$

And f_{nij} for Kelvin's solution and for Mindlin's solution (when source point does not belong to the semi-infinite surface) is given by:

$$f_{nij} = \frac{1-2\nu}{8\pi(1-\nu)} \left[\frac{n_j(\mathbf{P}) A_i(\theta) - n_i(\mathbf{P}) A_i(\theta)}{A^3(\theta)} \right] N_u^n(\gamma) J(\gamma) \quad (2.82)$$

The free term **c** for smooth boundary surface equals $\frac{1}{2}\mathbf{I}$ according to equation (2.36). For non smooth surface the free term **c** can be obtained following the Mantic procedure [9].

2.2.9 Boundary Element Method in a Two Dimensional Domain

3-D problems analysed by BEM have been the focus of previous sections for two reasons: (i) 3D BEM is more general than 2D BEM and most of 2D problems can be analysed using 3D BEM. (ii) Flac^{3D} software that is coupled with a 3D BEM program analyzes 3D problems (as its

name suggests); however, Flac^{3D} analyses 2D problems as special cases. Two dimensional domains are bounded by one dimensional boundary. The boundary should be subdivided into one dimensional isoparametric linear or higher order elements. Kelvin's fundamental solutions in an infinite medium for a two dimensional plain strain cases because of unit load applied in i direction are given by [5]:

$$
\begin{aligned}
U_{ij} &= \frac{-1}{8\pi(1-\nu)G}\{(3-4\nu)\delta_{ij}\ln r - r_{,i}r_{,j}\} \\
T_{ij} &= \frac{-1}{4\pi(1-\nu)r}\left\{\left[(1-2\nu)\delta_{ij} + 2r_{,i}r_{,j}\right]r_{,k}n_{,k} - (1-2\nu)(r_{,i}n_{j} - r_{,j}n_{i})\right\}
\end{aligned}
\tag{2.83}
$$

Melan's displacement U^{c}_{ki} and stress σ^{c}_{ijk} fundamental solutions (complementary part) in semi-infinite medium for two dimensional plain strain cases because of unit load applied in x, y or z direction (see Figure 2.17) are given as presented by Telles & Brebbia [10] as:

$$
\begin{aligned}
U^{c}_{xx} &= K_d\left\{-\left[8(1-\nu)^2-(3-4\nu)\right]\ln R + \frac{\left[(3-4\nu)r_x^2 + 2c\bar{y}\right]}{R^2} - \frac{4c\bar{y}r_x^2}{R^4}\right\} \\
U^{c}_{xy} &= K_d\left\{\frac{(3-4\nu)r_y r_x}{R^2} - \frac{4c\bar{y}R_y r_x}{R^4} + 4(1-\nu)(1-2\nu)\theta\right\} \\
U^{c}_{yx} &= K_d\left\{\frac{(3-4\nu)r_y r_x}{R^2} + \frac{4c\bar{y}R_y r_x}{R^4} - 4(1-\nu)(1-2\nu)\theta\right\} \\
U^{c}_{yy} &= K_d\left\{-\left[8(1-\nu)^2-(3-4\nu)\right]\ln R + \frac{\left[(3-4\nu)R_y^2 - 2c\bar{y}\right]}{R^2} + \frac{4c\bar{y}R_y^2}{R^4}\right\} \\
\sigma^{c}_{xxx} &= -K_s r_x\left\{\frac{3(1-2\nu)}{R^2} + \frac{2\left[r_x^2 - 4c\bar{y} - 2c^2 - 2\bar{y}R_y(1-2\nu)\right]}{R^4} + \frac{16c\bar{y}R_y^2}{R^6}\right\} \\
\sigma^{c}_{yxx} &= -K_s\left\{\frac{(3\bar{y}+c)(1-2\nu)}{R^2} + \frac{2\left[(2c\bar{y}+r_x^2)R_y - 2\bar{y}R_y^2(1-2\nu)\right]}{R^4} - \frac{16c\bar{y}R_y r_x^2}{R^6}\right\} \\
\sigma^{c}_{yyx} &= -K_s r_x\left\{\frac{(1-2\nu)}{R^2} - \frac{2\left[c^2 - \bar{y}^2 + 6c\bar{y} - 2\bar{y}R_y(1-2\nu)\right]}{R^4} + \frac{16c\bar{y}r_x^2}{R^6}\right\}
\end{aligned}
\tag{2.84}
$$

$$\sigma^{c}_{xxy} = -K_s\left\{\frac{(\bar{y}+3c)(1-2\nu)}{R^2}+\frac{2\left[R_y(r_x^2+2c^2)-2cr_x^2+2\bar{y}r_x^2(1-2\nu)\right]}{R^4}+\frac{16c\bar{y}R_y r_x^2}{R^6}\right\}$$

$$\sigma^{c}_{yxy} = -K_s r_x\left\{-\frac{(1-2\nu)}{R^2}+\frac{2\left[\bar{y}^2-2c\bar{y}-c^2+2\bar{y}R_y(1-2\nu)\right]}{R^4}+\frac{16c\bar{y}R_y^2}{R^6}\right\}$$

$$\sigma^{c}_{yyy} = -K_s\left\{\frac{(3\bar{y}+c)(1-2\nu)}{R^2}+\frac{2R_y(R_y^2+2c\bar{y})-4\bar{y}r_x^2(1-2\nu)}{R^4}-\frac{16c\bar{y}R_y r_x^2}{R^6}\right\}$$

$$K_s=\frac{1}{[4\pi(1-\nu)]}, K_d=\frac{1}{[8\pi G(1-\nu)]}$$

Elemental coefficients $\Delta\mathbf{U}^{e}_{ni}$ and $\Delta\mathbf{T}^{e}_{ni}$ are obtained numerically in a similar way as in a 3D analysis. Special care should be taken when source point is located close to or in the region of integration. The element in these cases should be subdivided into a number of sub regions or sub elements in similar techniques as used and explained before.

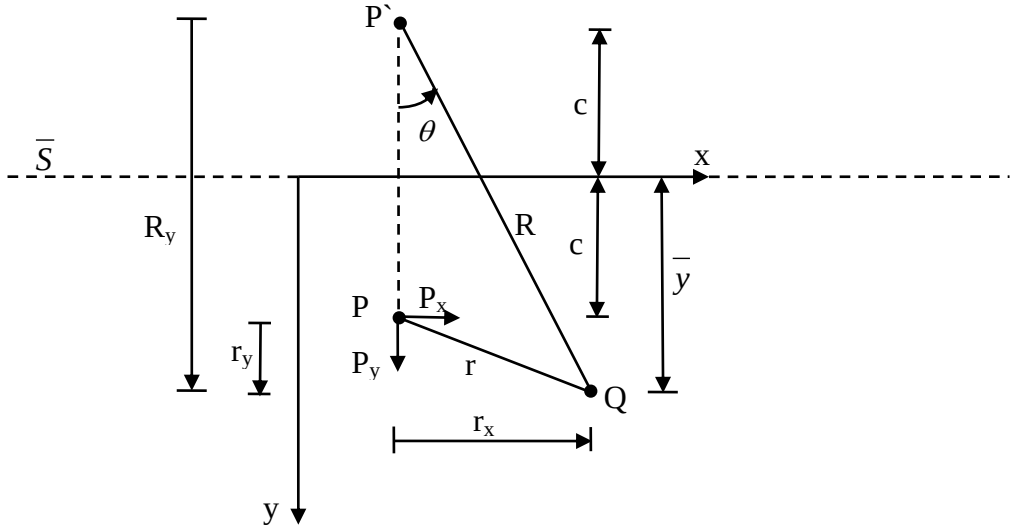


Figure 2.17 Unit Point Load Applied within Half Plane [10] /Melan's Solution/

2.2.10 Computation of the Stress inside a Semi-infinite Domain

Kernels **S** and **R** in equation (2.67), required to obtain the stress components at internal points inside the semi-infinite 2D (Melan) and 3D (Mindlin) domains, are a combination between Klevin's part tensors defined in equations (2.68) and the complementary part tensors, written in the general tensor notation form as [10]:

$$S^{c}_{ijk} = G\left(\frac{\partial U^{c}_{ik}}{\partial x_j} + \frac{\partial U^{c}_{jk}}{\partial x_i}\right) + \frac{2G\nu}{1-2\nu}\left(\frac{\partial U^{c}_{lk}}{\partial x_l}\right)\delta_{ij}$$

$$R^{c}_{ijk} = G\left[\frac{\partial \sigma^{c}_{kmi}}{\partial x_j} + \frac{\partial \sigma^{c}_{kmj}}{\partial x_i} + \frac{2\nu}{1-2\nu}\frac{\partial \sigma^{c}_{kml}}{\partial x_l}\delta_{ij}\right]n_m \qquad (2.85)$$

Although the derivatives of Melan fundamental solutions, seen in equations (2.85), are obtained and explicitly shown by Telles and Brebbia [10], to the author's knowledge there are no full and explicit $\mathbf{S^c}$ and $\mathbf{R^c}$ expressions based on Mindlin's fundamental solutions that have been derived or shown in any known literature. With the assistance of the Matlab 7.1 program, the author obtained the Mindlin's solutions derivatives and coded the complementary part tensors $\mathbf{S^c}$ and $\mathbf{R^c}$ into FORTRAN subroutines as well as C^{++} functions (see C++ code in Appendix f, pp. 352-372). The stress complementary tensors based on Melan's solutions, shown in reference [10], are also coded by the author into FORTRAN subroutines and C^{++} functions each (see C++ code in Appendix f, pp. 346-352). Moreover, Mindlin and Melan's fundamental solutions are coded by the author using the same programming languages, C++ and FORTRAN (see C++ code in Appendix f, pp.341-346).

Chapter 3

Fast Lagrangian Analysis of Continua in 3 Dimensions (Flac^{3D}) /Finite Difference Numerical Method/

3.1 Introduction

Flac^{3D} is the program intended to be coupled with the boundary element method explained earlier. The program name (Flac^{3D}) is an abbreviation of the phrase Fast Lagrangian Analysis of Continua in 3 Dimensions. It is described as Lagrangian analysis because the mechanical response (stress) is obtained or related back to the initial configuration. Flac^{3D} as described in its manual is "an explicit finite difference program to numerically study the mechanical behaviour of a continuous three-dimensional medium as it reaches equilibrium or steady plastic flow" [29]. Partial differential equations (Equilibrium equations and equations of motion) are discretized by Flac^{3D} using the finite difference numerical method which approximates the space and time derivatives of a variable by assuming linear variations of the variable over finite space and time intervals. Discontinuities in the continuum are studied by Flac^{3D} only as special cases such as interfaces or joints.

All of what has been detailed below (in chapter 3) is referred to Flac^{3D} manual, versions 2.00 & 3.00. The manual's notations are used in this chapter.

3.2 Mathematical Definitions [28]

- Cauchy's equations of motion based on the linear momentum principle, equivalent to Newton's second law of motion, are given by:

$$\sigma_{ji,j} + \rho b_i = \rho \frac{d\upsilon_i}{dt} \qquad (3.1)$$

Where ρ is the mass per unit volume of the medium, [b] is the body force per unit mass, and d[υ]/dt is the material derivative of the velocity. The residual force F_i at a node in the non linear

analysis is the difference between internal and external applied forces. Flac^{3D} considers the inertial force $\rho\frac{d\upsilon_i}{dt}$ in equation (3.1) as a residual to be reduced to zero iteratively if the continuum state of static equilibrium is achieved. However, the residual reaches a constant non negligible value if the continuum is in a steady (plastic) flow state. In static equilibrium, equation (3.1) is reduced to equation (2.3)**.**

- Strain-rate or rate of deformation is the symmetric part of velocity gradient (L):

$$\xi_{ij} = \frac{1}{2}(L + L^T) = \frac{1}{2}(\upsilon_{i,j} + \upsilon_{j,i}) \tag{3.2}$$

Where: $L = \upsilon_{i,j} = \frac{\partial \upsilon_i}{\partial x_j}$, **x** (**X**, t) is the current position vector and **X** is the initial position vector.

The first invariant of strain rate tensor is the rate of dilation.

- Material spin or rate of rotation tensor is the anti symmetric part of velocity gradient given by:

$$\omega_{ij} = \frac{1}{2}(L - L^T) = \frac{1}{2}(\upsilon_{i,j} - \upsilon_{j,i}) \tag{3.3}$$

- Constitutive equations are given by:

$$\breve{\sigma}_{ij} = H_{ij}(\sigma_{ij}, \xi_{ij}) \tag{3.4}$$

[H] is a given function and $[\breve{\sigma}]$ is the co-rotational Jaumann stress-rate. $[\breve{\sigma}]$ equals the material derivative of stress $\frac{d\sigma_{ij}}{dt}$ as seen by an observer on a Jaumann frame of reference rotating with a spin equal to the material spin that is defined by equation (3.3). $[\breve{\sigma}]$ is given by :

$$\breve{\sigma}_{ij} = \frac{d\sigma_{ij}}{dt} - \omega_{ik}\sigma_{kj} + \sigma_{ik}\omega_{kj} \tag{3.5}$$

3.3 Numerical Formulations [28]

Flac^{3D} Numerical solution is based on three main approaches as quoted from the Flac^{3D} manual:

1-"Finite difference approach: first–order space and time derivatives of a variable are approximated by finite difference, assuming linear variations of the variable over finite space and time intervals, respectively"[28]. The modeled medium is discretized into constant strain-rate tetrahedra elements (see Figure 3.1). The velocity gradient that appeared in equation (3.2) is computed for each element relying on the assumption of linear variation of velocity over the element.

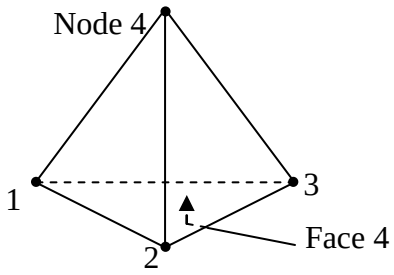


Figure 3.1 Tetrahedron [29]

2-"Discrete-model approach: The continuous medium is replaced by a discrete equivalent one in which all forces involved (applied and interactive) are concentrated at the nodes of a three-dimensional mesh used in the medium representation."

3-"Dynamic–solution approach: the inertial terms in the equations of motion are used as numerical means to reach the equilibrium state of the system under consideration."

Newton's law in equation (3.1) is discretized based on the last two approaches and solved numerically using finite difference in time.

3.4 Grid Discretization [28]

Flac^{3D} discretizes the medium, using a mixed discretization technique. The first discretization process is done following the user's guidelines into a coarse mesh of zones while the second

discretization process is done automatically by Flac^{3D} itself. The zones in the second process are discretized into one or two layers of tetrahedra (number of tetrahedra is 5 for each zone per one layer) as shown in Figure 3.2.

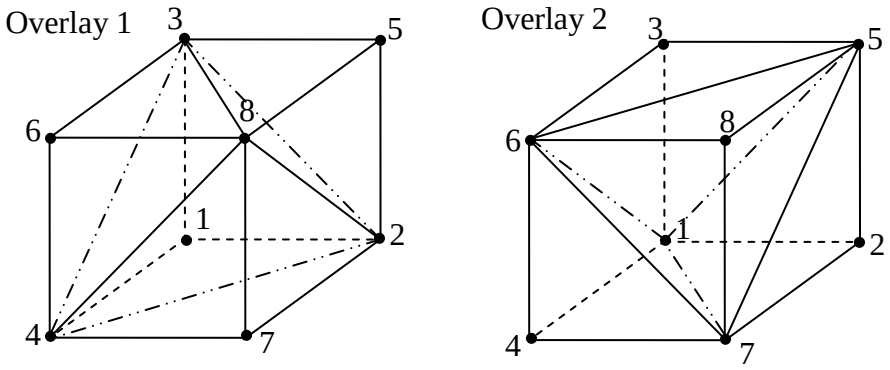


Figure 3.2 an 8-node zone with 2 overlays of 5 tetrahedra in each overlay [29]

The lack of volumetric deformation flexibility is caused by the over-stiff response of constant strain-rate tetrahedron. Some plasticity problems, for certain constitutive models, demand deformation with no volumetric strain. Mixed discretization provides volumetric flexibility by adjusting the strain-rate first invariant of a tetrahedron. The strain-rate of a tetrahedron locally numbered as [*l*] may be decomposed into deviatoric and volumetric parts as:

$$\xi_{ij}^{[l]} = \eta_{ij}^{[l]} + \frac{\xi^{[l]}}{3}\delta_{ij} \tag{3.6}$$

Where $\eta_{ij}^{[l]}$ is the deviatoric strain-rate tensor and $\xi^{[l]}$ is the strain-rate first invariant given by:

$$\xi^{[l]} = \xi_{ii}^{[l]} \tag{3.7}$$

The adjusted strain-rate of the tetrahedron is given by:

$$\xi_{ij}^{[l]} = \eta_{ij}^{[l]} + \frac{\xi^{z}}{3}\delta_{ij} \tag{3.8}$$

The zone first strain-rate invariant ξ^z is calculated as a volumetric average of first strain-rate invariant over all tetrahedra in the zone:

$$\xi^z = \frac{\sum_{l=1}^{5} \xi^{[l]} V^{[l]}}{\sum_{l=1}^{5} V^{[l]}} \tag{3.9}$$

Where: $V^{[l]}$ is the tetrahedron volume locally numbered as *l* and $\xi^{[l]}$ is given by eq. (3.7).

Although the individual tetrahedron violates the incompressibility of plastic flow (a required condition in some constitutive models), the zone does not. Having two layers of tetrahedra serves two objectives. First, it helps checking the symmetry of the solution for symmetric problems; and second, it helps obtaining a more accurate solution in the regions of high stress and deformation gradients. Nodal forces are computed by averaging the values obtained by each layer of tetrahedra. A similar procedure detailed for strain-rate first invariant is followed to produce constant mean stress after the occurrence of yielding for a zone.

3.5 Numerical Computations Algorithm [28]

The user assigns the initial and boundary conditions so Flac^{3D} proceedes its computations, after generating mixed grid discretization as outlined in the following steps:

1- Surface tractions and body forces are transformed into equivalent nodal forces and prescribed displacements are applied as nodal velocities.

2- Computations are repeated for each time step. Total number of time steps are either assigned by the user or if the command *solve* is used, the iteration will continue until the maximum out-of-balance force approaches zero by reaching the equilibrium state, or this force approaches a constant value indicating steady plastic flow state.

3- The strain rate tensor for each constant strain-rate tetrahedron is computed by:

$$\xi_{ij} = -\frac{1}{6V}\sum_{l=1}^{4}\left(\upsilon_i^l n_j^{(l)} + \upsilon_j^l n_i^{(l)}\right)\cdot S^{(l)} \tag{3.10}$$

Where: superscript l denotes tetrahedron node (see Figure 3.1), (l) is the face of tetrahedron opposite to the node l, $\mathbf{n}^{(l)}$ unit outward normal vector on the tetrahedron face (l), and $S^{(l)}$ is area of face (l). The above equation is obtained from equation (3.2) after computing the velocity gradients or spatial derivatives using finite difference approximation by applying Gauss theorem as the following:

$$\int_V \nu_{i,j} dV = \int_S \nu_i n_j dS \tag{3.11}$$

Because the variation of velocity over the tetrahedron is assumed to be linear, the above integrations can be written as:

$$V\nu_{i,j} = \sum_{f=1}^{4} \overline{\nu}_i^{(f)} n_j^{(f)} S^{(f)} \tag{3.12}$$

$\overline{\nu}_i^{(f)}$ is the average value of velocity component i on the face f of the tetrahedron:

$$\overline{\nu}_i^{(f)} = \frac{1}{3}\sum_{l=1, l\neq f}^{4} \nu_i^l \tag{3.13}$$

Finite difference approximation of velocity gradients, after substituting equation (3.13) into equation (3.12) becomes:

$$\nu_{i,j} = -\frac{1}{3V}\sum_{l=1}^{4} \nu_i^l n_j^{(l)} S^{(l)} \tag{3.14}$$

Again, substituting equation (3.14) with equation (3.2) produces equation (3.10).

Diagonal components of the strain-rate tensor for each tetrahedron, after computing the strain rate for every tetrahedron of a zone, are adjusted using equations (3.6) to (3.9).

4- Stress increment for each tetrahedron is computed using constitutive equation (3.4):

$$\Delta\breve{\sigma}_{ij} = H_{ij}(\sigma_{ij}, \Delta\varepsilon_{ij}) \tag{3.15}$$

Strain increment $\Delta\varepsilon_{ij}$ is given using equation (3.10) by:

$$\Delta\varepsilon_{ij} = -\frac{\Delta t}{6V}\sum_{l=1}^{4}\left(\upsilon_i^l n_j^{(l)} + \upsilon_j^l n_i^{(l)}\right)\cdot S^{(l)} \tag{3.16}$$

Where: Δt is a time step. Stress increment is corrected using equation (3.5) by:

$$\Delta\sigma_{ij} = \Delta\breve{\sigma}_{ij} + (\omega_{ik}\sigma_{kj} - \sigma_{ik}\omega_{kj})\Delta t \tag{3.17}$$

Where: The material spin or rate of rotation ω_{kj} is given by equation (3.3). The stress correction is neglected if small-strain mode is used which is the default mode in Flac^{3D}. Tetrahedron stress is adjusted in a similar method followed for the strain rate explained before. Stress for a tetrahedron *l* is given by:

$$\sigma_{ij}^{[l]} = s_{ij}^{[l]} + \sigma^z\delta_{ij} \tag{3.18}$$

Where: $s_{ij}^{[l]}$ is a deviatoric stress tensor and σ^z is the zone's first invariant, σ^z is calculated as the volumetric average value over all tetrahedra in the zone:

$$\sigma^z = \frac{\sum_{l=1}^{5}\sigma^{[l]}V^{[l]}}{\sum_{l=1}^{5}V^{[l]}} \tag{3.19}$$

$\sigma^{[l]}$ is the mean or hydrostatic stress of tetrahedron locally labelled as *l* given by:

$$\sigma^{[l]} = \frac{1}{3}\sigma_{ii}^{[l]} \tag{3.20}$$

5- According to the second and third approaches of numerical formulations section, the medium mass is discretized and redistributed by computing the tetrahedron contribution of the nodal mass by using the following equation:

$$m^{l} = \frac{\alpha_1}{9V^{[l]}} \max\left(\left[n_i^{(l)} S^{(l)} \right]^2, i = 1,3 \right) \tag{3.21}$$

Where: $\alpha_1 = K + 4/3G$, K is bulk modulus and G is shear modulus.

Equation (3.21) is obtained by first assuming the medium as an assembly of point masses located at the nodes and connected by linear springs. The critical time step of a single mass-spring system governed by second order differential equation $\left(-kx = m\frac{d^2x}{dt^2} \right)$ is given:

$$\Delta t = \frac{T}{\pi} \tag{3.22}$$

T is the period of the system given by:

$$T = 2\pi\sqrt{\frac{m}{4k}} \tag{3.23}$$

Where: 4k is the equivalent stiffness of a single mass-spring system connected in series to similar systems. In a stable system the mass should satisfy the condition:

$$m \geq k(\Delta t)^2 \tag{3.24}$$

Condition 3.24 is derived from equations (3.22) and (3.23)

For uniform unit time step ($\Delta t = 1$) the nodal mass m^{l} should be greater or equal to the nodal stiffness contribution k^{l}. Spring reaction force $-k_{ij}^{l} u_j^{l}$ equals the tetrahedron internal force contribution at node *l* that equals ($T_i^{l}/3$) as shown in equation (3.40). Internal force T_i^{l} is given by:

$$T_i^{l} = \sigma_{ij} n_j^{(l)} S^{(l)} \tag{3.25}$$

This equation will be obtained in the next computation step. Taking the variation over time of equation (3.25) and using the statement ($-k_{ij}^{l} u_j^{l} = T_i^{l}/3$) mentioned before will give:

$$\frac{d\sigma_{ij}}{3} n_j^{(l)} S^{(l)} = -k_{ij}^{l} \upsilon_j^{l} dt \tag{3.26}$$

If unit velocity is applied in q direction at node *l*, and zero at the other directions, diagonal terms of local stiffness matrix in equation (3.26) becomes (3.27) by applying Hook's law:

$$k_{qq} dt = -\frac{\xi_{qq} \alpha_1 dt}{3} n_q^{(l)} S^{(l)} \tag{3.27}$$

Using equation (3.10) for a unit nodal velocity, the strain-rate is given by:

$$\xi_{qq} = -\frac{1}{3V} n_q^{(l)} S^{(l)} \tag{3.28}$$

If equation (3.28) is substituted into equation (3.27) we get:

$$k_{qq} = \frac{\alpha_1}{9V} \left[n_q^{(l)} S^{(l)} \right]^2 \tag{3.29}$$

The maximum nodal stiffness at node *l* is:

$$k^{l} = \max(k_{11}, k_{22}, k_{33}) \tag{3.30}$$

Equations (3.24), (3.29) and (3.30) lead to equation (3.21) for unity time step.

6- The out of balance force is computed by first obtaining the tetrahedron contribution at node locally numbered as *l* by the equation:

$$p_i^{l} = \frac{1}{3} \sigma_{ij} n_j^{(l)} S^{(l)} + \frac{1}{4} \rho b_i V \tag{3.31}$$

Contributions of all tetrahedra that share the node globally numbered as $< l >$ is summed. This summation operation is given the symbol $[[p_i]]^{<l>}$. Out of balance force as the difference of external applied force $P_i^{<l>}$ at node $< l >$ and the internal force $[[p_i]]^{<l>}$ is:

$$F_i^{<l>} = [[p_i]]^{<l>} + P_i^{<l>} \tag{3.32}$$

Equations (3.31), (3.32) and (3.25) are proved to be true by the following:

The applied nodal forces $[f]^l$ (l =1, 4) acting on a tetrahedron are in a static equilibrium with the tetrahedron stress and equivalent body force. External work-rate done by the nodal forces $[f]^l$ and body forces according to the principal of virtual work equals the internal work-rate done by stress over virtual linear velocity field $\delta[\upsilon]^l$. External work is given by:

$$E = \sum_{l=1}^{4} \delta\upsilon_i^l f_i^l + E^b + E^I \tag{3.33}$$

E^b The external work-rate of body forces given by $E^b = \rho b_i \int_V \delta\upsilon_i dV$ can be written in a finite difference formulation as: $E^b = \sum_{l=1}^{4} \delta\upsilon_i^l \frac{\rho b_i V}{4}$ (3.34)

E^I The external work-rate of inertial forces given by $E^I = -\int_V \rho\delta\upsilon_i \frac{d\upsilon_i}{dt} dV$ can be written in a finite difference formulation as:

$$E^I = -\sum_{l=1}^{4} \delta\upsilon_i^l m^l \left(\frac{d\upsilon_i}{dt}\right)^l \tag{3.35}$$

Internal work-rate done by stress given by $I = \int_V \delta\xi_{ij}\sigma_{ij} dV$ can be written in a finite difference formulation for a constant strain-rate using equation (3.10) as:

$$I = -\frac{1}{6}\sum_{l=1}^{4} \left(\delta\upsilon_i^l \sigma_{ij} n_j^{(l)} + \delta\upsilon_j^l \sigma_{ij} n_i^{(l)}\right) \cdot S^{(l)} \tag{3.36}$$

Because of stress, tensor symmetry equation (4.36) is rewritten as the following:

$$I = -\frac{1}{3}\sum_{l=1}^{4} \delta\upsilon_i^l T_i^l \tag{3.37}$$

Where T_i^l is the internal force due to stress effect at node l given by equation (3.25):

$$T_i^l = \sigma_{ij} n_j^{(l)} S^{(l)}$$

If equations (3.34) and (3.35) are substituted in equation (3.33) and if the resulted equation is equated with equation (3.37), we will have for each node *l*:

$$-f_i^l = \frac{T_i^l}{3} + \frac{\rho b_i V}{4} - m^l \left(\frac{d\upsilon_i}{dt} \right)^l \tag{3.38}$$

According to the previous definitions and knowing that nodal force is equal to the external applied force $P_i^{<l>}$ at node < *l* >, out of balance force $F_i^{<l>}$ at a node globally numbered as < *l* > shared by number of tetrahedra is given by:

$$F_i^{<l>} = M^{<l>} \left(\frac{d\upsilon_i}{dt} \right)^{<l>} \quad l = 1, n_n \tag{3.39}$$

$$F_i^{<l>} = [[\frac{T_i^l}{3} + \frac{\rho b_i V}{4}]] + P_i^{<l>} \tag{3.40}$$

Where: n_n is the total number of nodes involved in the medium representation and $M^{<l>}$ is the nodal mass given by: $M^{<l>} = [[m]]^{<l>}$ (3.41)

The out of balance force is monitored by Flac^{3D} to indicate the state of equilibrium if it reached zero value or close to zero and to indicate the state of steady flow if it reached constant value; otherwise, no equilibrium is taking place.

7- Compute mechanical damping force (will be added to the out of balance force in the next step) by the following equation:

$$F_i^{<l>} = -\alpha . \left| F_i^{<l>} \right| . \text{sign} \left(\upsilon_i^{<l>} \right) \tag{3.42}$$

Where: $\text{sign}\left(\upsilon_i^{<l>}\right) = \begin{cases} +1, & \text{if} : \upsilon_i^{<l>} > 0; \\ -1, & \text{if} : \upsilon_i^{<l>} < 0; \\ 0, & \text{if} : \upsilon_i^{<l>} = 0 \end{cases}$

α is constant given by Flac^{3D} the value of 0.8.

8- New nodal velocity is computed using central finite formulation in time and equations (3.39) and (3.42) as:

$$\upsilon_i^{<l>}\left(t+\frac{\Delta t}{2}\right)=\upsilon_i^{<l>}\left(t-\frac{\Delta t}{2}\right)+\frac{\Delta t}{M^{<l>}}\left(F_i^{<l>}+F_i^{<l>}\right) \tag{3.43}$$

9- Nodal displacement is computed by:

$$u_i^{<l>}(t+\Delta t)=u_i^{<l>}(t)+\Delta t.\upsilon_i^{<l>}\left(t+\frac{\Delta t}{2}\right) \tag{3.44}$$

10- The geometry is updated if the large strain mode is chosen by the user using the equation:

$$x_i^{<l>}(t+\Delta t)=x_i^{<l>}(t)+\Delta t.\upsilon_i^{<l>}\left(t+\frac{\Delta t}{2}\right) \tag{3.45}$$

If the small strain mode is used, which is the default mode, no geometry update is done.

11- Steps 3 to 10 are repeated until a state of equilibrium or steady flow is reached; or else, a state of non equilibrium is indicated (see Figure 3.a in Appendix a, p. 289).

3.6 Constitutive Models [29]

Flac^{3D} is one of the richest programs of constitutive models. It has twelve constitutive models that can be classified into three main categories: null, elastic and plastic model groups.

3.6.1 Incremental Equations of the Theory of Plastic Flow

It has been mentioned in step 4 in the numerical computations algorithm that the stress increment is obtained by a very general form of constitutive model in equation (3.15). What will be detailed in this section is how to obtain the new stress at time t+Δt by having the stress value at time t and the strain increment for time step Δt. Equations of failure criterion, total strain and stress increments, and flow rule as presented in the Flac^{3D} manual are rewritten as the following:

1- Failure criteria: $$f(\underline{\sigma}_n+\Delta\underline{\sigma}_n)=0 \tag{3.46}$$

Where: $[\underline{\sigma}]$ is the generalized stress vector with components $\underline{\sigma}_i$ (i=1, n).

2- Total strain increment: $\Delta\underline{\varepsilon}_i = \Delta\underline{\varepsilon}_i^e + \Delta\underline{\varepsilon}_i^p$ (3.47)

Where: $[\underline{\varepsilon}]$ is the generalized strain vector with components $\underline{\varepsilon}_i$ (i=1, n).

3- Stress increment as linear function S_i of elastic strain:

$$\Delta\underline{\sigma}_i = S_i(\Delta\underline{\varepsilon}_n^e) \qquad i=1, n \tag{3.48}$$

4- Non associated flow rule:

$$\Delta\underline{\varepsilon}_i^p = \lambda \frac{\partial g}{\partial \underline{\sigma}_i} \tag{3.49}$$

If the plastic potential function g is the same as yield function f, the above flow rule is called an associated flow rule.

Substituting equations (3.47) and (3.49) into equation (3.48) and taking into consideration the linearity of function S_i gives:

$$\Delta\underline{\sigma}_i = S_i(\Delta\underline{\varepsilon}_n) - \lambda S_i\left(\frac{\partial g}{\partial \underline{\sigma}_n}\right) \tag{3.50}$$

For the special case of linear failure criterion f, equation (3.46) can be rewritten as:

$$f(\underline{\sigma}_n) + f^*(\Delta\underline{\sigma}_n) = 0 \tag{3.51}$$

Where: f^* is the function f without its constant term:

$$f^*(.) = f(.) - f(\underline{0}_n) \tag{3.52}$$

When stress vector $\underline{\sigma}_n$ touches yield surface in the stress space we have $f(\underline{\sigma}_n) = 0$ and after substituting (4.50), equation (3.51) becomes:

$$f^*\left[S_n(\Delta\underline{\varepsilon}_n)\right] - \lambda f^*\left[S_n\left(\frac{\partial g}{\partial \underline{\sigma}_n}\right)\right] = 0 \tag{3.53}$$

New stress components and initial elastic stress guess are given by:

$$\underline{\sigma}_i^N = \underline{\sigma}_i + \Delta\underline{\sigma}_i \tag{3.54}$$

$$\underline{\sigma}_i^I = \underline{\sigma}_i + S_i(\Delta\underline{\varepsilon}_n) \tag{3.55}$$

Stress increment $\Delta\underline{\sigma}_i = S_i(\Delta\underline{\varepsilon}_n)$ term of initial elastic stress guess in equation (3.55) is attributed completely to the total stain increment $\Delta\underline{\varepsilon}_n$ by assuming that no increment of plastic strain is taking place. Initial elastic stress guess will be corrected by the following procedure:

Using equation (3.55), equation (3.51) and by having $f(\underline{\sigma}_n) = 0$ we get:

$$f(\underline{\sigma}_n^I) = f^*[S_n(\Delta\underline{\varepsilon}_n)]$$

From the above equation and from equations (3.53) and (3.52) the plastic multiplier λ is obtained by:

$$\lambda = \frac{f(\underline{\sigma}_n^I)}{f[S_n(\partial g/\partial\underline{\sigma}_n)] - f(\underline{0}_n)} \tag{3.56}$$

Substituting the stress increment defined in equation (3.50) into equation (3.54) and using the initial elastic stress guess definition in equation (3.55) gives the corrected new stress components by:

$$\underline{\sigma}_i^N = \underline{\sigma}_i^I - \lambda S_i\left(\frac{\partial g}{\partial\underline{\sigma}_n}\right) \tag{3.57}$$

Flac^{3D} calculates elastic stress guess $\underline{\sigma}_i^I$ at time t+Δt by adding the elastic stress increment to the stress value at time t using equation (3.55) because the elastic stress increment is a linear function of the total strain. If yield function of the elastic stress violated the yield criterion $(f(\underline{\sigma}_n^I) \geq 0)$, the guess will be corrected using equation (3.57) after obtaining λ from equation (3.56). If the stress state is located below the yield surface $(f(\underline{\sigma}_n^I) < 0)$, the elastic stress guess gives the new stress value $\underline{\sigma}_i^N$ at time t+Δt (see Figure 3.b in Appendix a, p. 290).

3.6.2 Null Model Group

The stress in the null region (excavated material) is set automatically to zero:

$$\sigma_{ij}^{N} = 0 \tag{3.58}$$

The removed material can be filled later by another material using a different model.

3.6.3 Elastic Model Group

There are three elastic models in Flac^{3D} to represent reversible path-independent deformation. These models are the isotropic, orthotropic, and transversely isotropic model.

3.6.3.1 The Elastic Isotropic Model

Hooke's law is the governing stress-strain relationship in this model similar to equation (2.2) rewritten in incremental form as:

$$\Delta\sigma_{ij} = 2G\Delta\varepsilon_{ij} + \alpha_2 \Delta\varepsilon_{kk}\,\delta_{ij} \tag{3.59}$$

α_2 is Lame's constant (λ) written in terms of bulk modulus K and shear modulus G as:

$$\alpha_2 = \lambda = K - \frac{2}{3}G \tag{3.60}$$

New stress at time $t+\Delta t$ is obtained by:

$$\sigma_{ij}^{N} = \sigma_{ij} + \Delta\sigma_{ij} \tag{3.61}$$

3.6.3.2 The Elastic Orthotropic Model

This model is assumed to have three orthogonal planes of elastic symmetry. Incremental strain-stress relationship in local coordinates 1`, 2`and 3` normal to the orthogonal planes is given by:

$$\begin{Bmatrix} \Delta\varepsilon`_{11} \\ \Delta\varepsilon`_{22} \\ \Delta\varepsilon`_{33} \\ 2\Delta\varepsilon`_{12} \\ 2\Delta\varepsilon`_{13} \\ 2\Delta\varepsilon`_{23} \end{Bmatrix} = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_2} & -\frac{\nu_{13}}{E_3} & & & \\ -\frac{\nu_{21}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_3} & & 0 & \\ -\frac{\nu_{31}}{E_1} & -\frac{\nu_{32}}{E_2} & \frac{1}{E_3} & & & \\ & & & \frac{1}{G_{12}} & & \\ & & & & \frac{1}{G_{13}} & \\ & & 0 & & & \frac{1}{G_{23}} \end{bmatrix} \begin{Bmatrix} \Delta\sigma`_{11} \\ \Delta\sigma`_{22} \\ \Delta\sigma`_{33} \\ \Delta\sigma`_{12} \\ \Delta\sigma`_{13} \\ \Delta\sigma`_{23} \end{Bmatrix} \tag{3.62}$$

Because of the symmetry of the strain-stress matrix it is true that:

$$\frac{\nu_{21}}{E_1} = \frac{\nu_{12}}{E_2};\ \frac{\nu_{31}}{E_1} = \frac{\nu_{13}}{E_3} \text{ and } \frac{\nu_{32}}{E_2} = \frac{\nu_{23}}{E_3} \tag{3.63}$$

There are only nine independent constants as the above equation implies. These constants are:

E_1, E_2, E_3 Young's moduli in the directions of the local axes

G_{23}, G_{13}, G_{12} Shear moduli in planes parallel to the local coordinate planes.

$\nu_{23}, \nu_{13}, \nu_{12}$ Poisson's ratio where $\nu_{i`j`}$ represents lateral contraction in local direction i` caused by tensile stress in local direction j`.

The user prescribes the orientation of local axes in the global axes space. Stress increment as a linear function of strain increment in global coordinates is given by:

$$\Delta\{\sigma\} = [D]\Delta\{\varepsilon\} \tag{3.64}$$

And

$$[D] = [Q]^T [D`][Q] \tag{3.65}$$

Where [D`] is inverse of the strain-stress matrix in equation (3.62) and [Q] is the transformation matrix. New stress at time t+Δt is obtained by equation (3.61).

3.6.3.3 The Elastic Transversely Isotropic Model

This model is a special case of the Orthotropic Model assumed to have a 1`2` plane of isotropy. The normal axis on this plane is 3`. Incremental strain-stress relationship in local coordinates for this model is the same as equation (3.62) but by making:

$$E_1 = E_2 = E; \qquad E_3 = E`$$

$$\nu_{12} = \nu; \qquad \nu_{13} = \nu_{23} = \nu` \tag{3.66}$$

$$G_{12} = G = \frac{E_1}{2(1+\nu_{12})}; \qquad G_{13} = G_{23} = G`$$

There are only five independent constants as seen from equation (3.66). Orientation of the local axes is prescribed by the user. The stress increment in global coordinates is given by equation (3.64) and new stress is given by equation (3.61).

3.6.4 The Plastic Model Group

There are eight constitutive models in this group. Two of them will be explained in some details. These models are Drucker-Prager, Mohr-Coulomb models. Von Mises and Tresca yield criteria are considered as special cases of Drucker-Prager and Mohr-Coulomb models, respectively.

3.6.4.1 The Drucker-Prager Model

Failure in this model is controlled by Drucker-Prager criterion for shear failure with non-associated flow rule and by tension cutoff for tension failure with associated flow rule. The failure envelope f (τ, σ) = 0 is divided by this composite failure criteria into two parts (see Figure 3.3). The first part from A to B on the figure is defined by Drucker-Prager failure criterion ($f^s = 0$):

$$f^s = \tau + q_\phi \sigma - k_\phi$$

$$\tau = \sqrt{J_2} = \sqrt{\frac{1}{2} s_{ij} s_{ij}}\ ,\ \sigma = \frac{I_1}{3} = \frac{\sigma_{kk}}{3} \tag{3.67-a}$$

Where q_ϕ and k_ϕ are material constants. If the Drucker-Prager yield-cone surface is matched with the outer vertices of the Mohr-Coulomb yield surface, the constants q_ϕ and k_ϕ are given by:

$$q_\phi = \frac{6\sin\phi}{\sqrt{3}(3-\sin\phi)} \qquad ; \qquad k_\phi = \frac{6c\cos\phi}{\sqrt{3}(3-\sin\phi)} \tag{3.67-b}$$

If the Drucker-Pruger cone is matched with the inner vertices of the Mohr-Coulomb hexagon, the constants become:

$$q_\phi = \frac{6\sin\phi}{\sqrt{3}(3+\sin\phi)} \qquad ; \qquad k_\phi = \frac{6c\cos\phi}{\sqrt{3}(3+\sin\phi)} \tag{3.67-c}$$

The second part from B to C is defined by tension failure criterion $f^t = 0$ as:

$$f^t = \sigma - \sigma^t \tag{3.68}$$

Where: σ^t is the tensile strength. If q_ϕ equals zero, the default tensile strength is set to zero, but If q_ϕ is not zero, the maximum value of the tensile strength is given by:

$$\sigma^t_{max} = \frac{k_\phi}{q_\phi} \tag{3.69}$$

When $q_\phi = 0$ the Drucker-Prager criterion will transform into the Von-Mises criterion.

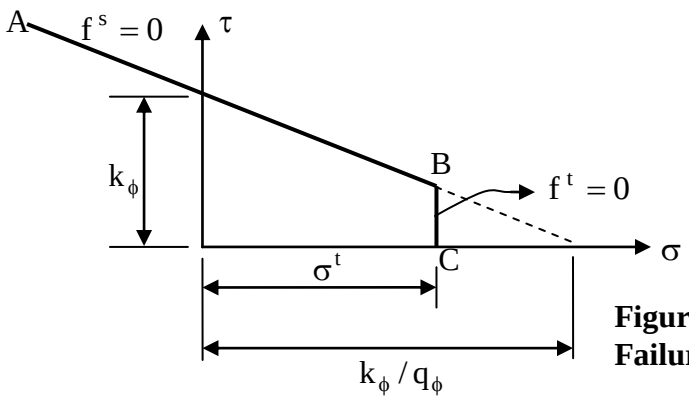


Figure 3.3 Flac^{3D} Drucker-Prager Failure Criterion [29]

Function h (τ, σ) = 0 represents the diagonal between $f^s = 0$ and $f^t = 0$ in τ, σ plane as seen in Figure 3.4 and given by:

$$h = \tau - \tau^P - a^P(\sigma - \sigma^t) \tag{3.70}$$

Where:

$$\tau^P = k_\phi - q_\phi \sigma^t$$
$$a^P = \sqrt{1 + q_\phi^2} - q_\phi \tag{3.71}$$

Elastic guess that violates the composite yield function and is represented by a point in τ, σ plane is located either in domain 1 or in domain 2 depending on h sign. If h is positive, the stress point is placed on $f^s = 0$ curve and shear failure occurs. If h is negative, the stress point is placed on $f^t = 0$ curve and tension failure occurs. Flow rule in shear failure is non-associated and plastic potential function is given by:

$$g^S = \tau + q_\psi \sigma \tag{3.72}$$

Where q_ψ is a constant and equals q_ϕ if the flow rule is associated. Flow rule in tensile failure is associated and its potential function is given by:

$$g^t = \sigma \tag{3.73}$$

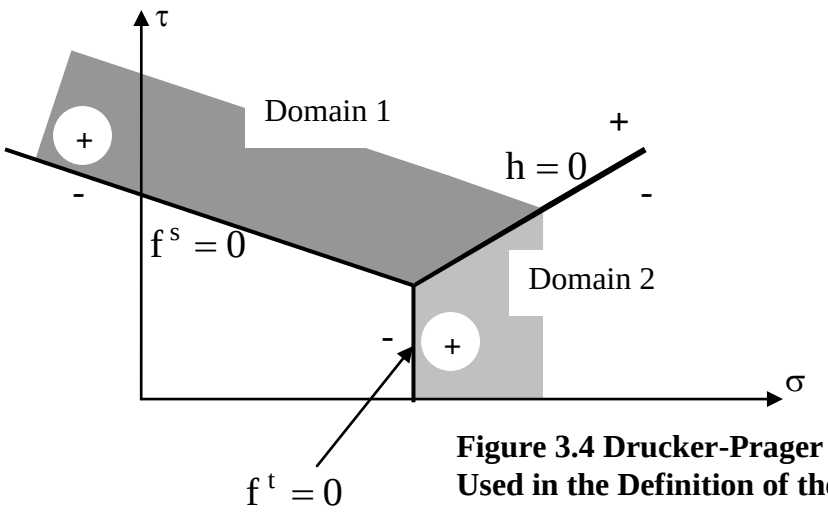


Figure 3.4 Drucker-Prager Model-Domains Used in the Definition of the Flow Rule

According to Hook's law equation (section 3.6.3**)**, the incremental form of tangential stress τ and normal stress σ may be written as:

$$\Delta\tau = S_1(\Delta\gamma^e, \Delta\varepsilon^e) = G\Delta\gamma^e$$
$$\Delta\sigma = S_2(\Delta\gamma^e, \Delta\varepsilon^e) = K\Delta\varepsilon^e \tag{3.74}$$

Where:

$$\Delta\gamma = \sqrt{2\Delta e_{ij}\Delta e_{ij}}$$
$$\Delta\varepsilon = \Delta\varepsilon_{kk} \tag{3.75}$$

Δe_{ij} is the incremental deviatoric-strain.

Partial differentiation of the potential functions is obtained, to compute the new stress value and the plastic correction at time t+Δt, from equation (3.72) first considering the shear failure case:

$$\frac{\partial g^s}{\partial\tau} = 1; \quad \frac{\partial g^s}{\partial\sigma} = q_\psi \tag{3.76}$$

Substituting the above differentiations for $\Delta\gamma^e$ and $\Delta\varepsilon^e$ in equation (3.74) gives:

$$S_1\left(\frac{\partial g^s}{\partial\tau}, \frac{\partial g^s}{\partial\sigma}\right) = G$$
$$S_2\left(\frac{\partial g^s}{\partial\tau}, \frac{\partial g^s}{\partial\sigma}\right) = Kq_\psi \tag{3.77}$$

We get from equations (3.57), (3.67) and (3.77) by having $f = f^s$:

$$\tau^N = \tau^I - \lambda^s G$$
$$\sigma^N = \sigma^I - \lambda^s K q_\psi \tag{3.78}$$

Where λ^s by using equation (3.56) is given by:

$$\lambda^s = \frac{f^s(\tau^I, \sigma^I)}{G + K q_\phi q_\psi} \tag{3.79}$$

The new value of deviatoric stress, according to the definition of tangential stress τ, is written, similar to τ^N in equation (3.78), as the following:

$$s_{ij}^N = s_{ij}^I - \lambda^s G \tag{3.80}$$

The new value of deviatoric stress tensor in equation (3.80), using equation (3.78) and the definition of τ, becomes:

$$s_{ij}^N = s_{ij}^I \frac{\tau^N}{\tau^I} \tag{3.81}$$

The new value of stress components, according to the definition of stress tensor and using equations (3.78) and (3.81), become:

$$\sigma_{ij}^N = s_{ij}^N + \sigma^N \delta_{ij} \tag{3.82}$$

Partial differentiation of plastic potential in equation (3.73) for the tensile failure case is given by:

$$\frac{\partial g^t}{\partial \tau} = 0 \; ; \quad \frac{\partial g^t}{\partial \sigma} = 1 \tag{3.83}$$

From equation (3.74) we have:

$$S_1\left(\frac{\partial g^t}{\partial \tau}, \frac{\partial g^t}{\partial \sigma}\right) = 0$$
$$S_2\left(\frac{\partial g^t}{\partial \tau}, \frac{\partial g^t}{\partial \sigma}\right) = K \tag{3.84}$$

By having $f = f^t$ from equations (3.57), (3.68) and (3.84), we get:

$$\tau^N = \tau^I$$
$$\sigma^N = \sigma^I - \lambda^t K \tag{3.85}$$

Where λ^t by using equation (3.56) is given by:

$$\lambda^t = \frac{\sigma^I - \sigma^t}{K} \tag{3.86}$$

By substituting (3.86) in equation (3.85) we get:

$$\tau^N = \tau^I \quad ; \quad \sigma^N = \sigma^t \tag{3.87}$$

We can conclude from equation (3.87) that in tensile failure case we have $s_{ij}^N = s_{ij}^I$, and from the stress tensor definition the following is obtained:

$$\sigma_{ij}^N = s_{ij}^I + \sigma^t \delta_{ij} \tag{3.88}$$

And

$$\sigma_{ij}^N = \sigma_{ij}^I + (\sigma^t - \sigma^I)\delta_{ij} \tag{3.89}$$

The computations algorithm that Flac^{3D} follows for this model is:

1- The elastic stress guess σ_{ij}^I is computed by applying Hook's law in section 3.6.3 to the total strain increment $\Delta\varepsilon_{ij}$ and then by adding the resulted stress increment to the old value of stress according to equation (3.55).

2- Deviatoric stress s_{ij}^I is obtained by having σ_{ij}^I computed in step one and τ^I, σ^I computed from equation (3.67).

3- Yield functions of τ^I, σ^I are obtained using equations (3.67) and (3.68). If the composite yield criterion is violated, function h (τ^I, σ^I) is computed.

4- If function h (τ^I, σ^I) > 0 shear failure is taking place, τ^N, σ^N are computed from equations (3.78) and (3.79).

5- The new deviatoric stress s_{ij}^{N} tensor is computed using equation (3.81) and new stress components σ_{ij}^{N} are computed using equation (3.82).

6- If function h $(\tau^{I},\sigma^{I}) \le 0$ tensile failure is taking place, τ^{N},σ^{N} are computed from equation (3.87). New stress components are computed using equation (3.89).

7- If the stress point τ^{I},σ^{I} is located below the composite failure envelope, the deformation is completely elastic. The elastic guess needs no correction and $\sigma_{ij}^{N} = \sigma_{ij}^{I}$.

3.6.4.2 Mohr-Coulomb Model

Failure in this model is governed by composite failure criterion build up by two failure criteria. The first criterion is Mohr-Coulomb for shear failure with non-associated flow rule. Shear failure envelope $f^{s}(\sigma_1,\sigma_3) = 0$ is represented in the plane σ_1,σ_3 by the curve from A to B (see Figure 3.5). The Mohr-Coulomb failure criterion is given by:

$$f^{s} = \sigma_1 - \sigma_3 N_{\phi} + 2c\sqrt{N_{\phi}} \tag{3.90}$$

The principal stresses are arranged as: $\sigma_1 \le \sigma_2 \le \sigma_3$ and constant N_{ϕ} is given by:

$$N_{\phi} = \frac{1+\sin\phi}{1-\sin\phi} \tag{3.91}$$

The second criterion is the tension cutoff for tensile failure with the associated flow rule. Tension failure envelope $f^{t}(\sigma_3,\sigma^{t}) = 0$ from B to C is defined by the tensile failure criterion given by:

$$f^{t} = \sigma_3 - \sigma^{t} \tag{3.92}$$

Tensile strength σ^{t} maximum value corresponds to the intersection point of line $f^{s} = 0$ and line $\sigma_1 = \sigma_3$ (see Figure 3.5) and given by:

$$\sigma^{t}_{max} = \frac{c}{\tan\phi} \tag{3.93}$$

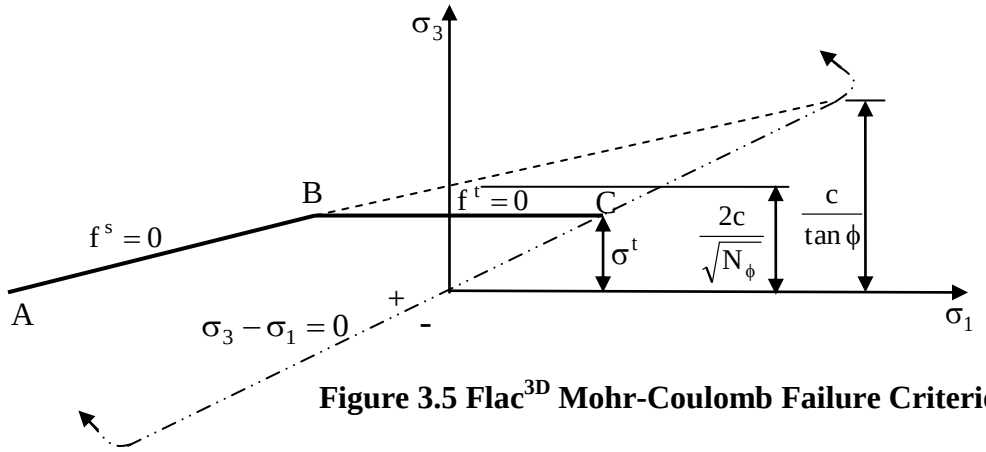


Figure 3.5 Flac^{3D} Mohr-Coulomb Failure Criterion [29]

If ϕ =0 constant N_ϕ equals 1, Mohr-Coulomb criterion transforms into Tresca criterion. Function $[\, h(\sigma_1, \sigma_3) = 0 \,]$ represents the diagonal between $f^s = 0$ and $f^t = 0$ in the plane σ_1, σ_3 (see Figure 3.6):

$$h = \sigma_3 - \sigma^t + a^P(\sigma_1 - \sigma^P) \tag{3.94}$$

Where:

$$a^P = \sqrt{1 + N_\phi^2} + N_\phi$$
$$\sigma^P = \sigma^t N_\phi - 2c\sqrt{N_\phi} \tag{3.95}$$

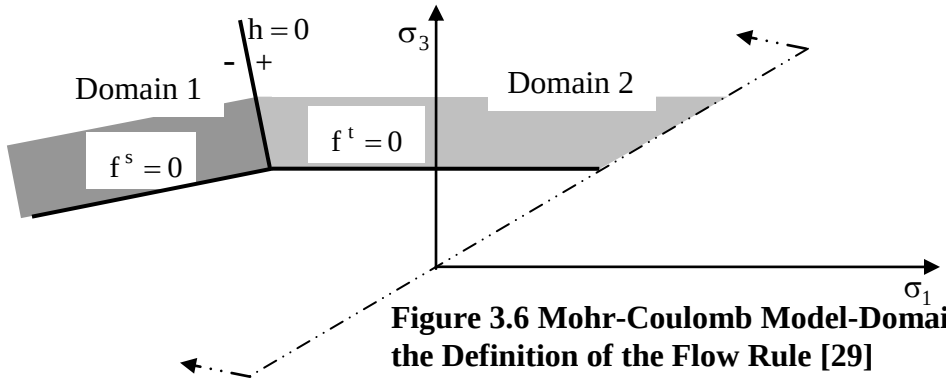


Figure 3.6 Mohr-Coulomb Model-Domains Used in the Definition of the Flow Rule [29]

If the elastic stress guess violated the composite failure criterion, function h is calculated. If h is negative, the stress point is placed on curve $f^s = 0$ and shear failure is declared. If h is positive, the stress point is placed on curve $f^t = 0$ and tension failure is declared.

Hook's law of generalized strain and stress incremental form may be rewritten as:

$$\begin{aligned}\Delta\sigma_1 &= S_1(\Delta\varepsilon_1^e, \Delta\varepsilon_2^e, \Delta\varepsilon_3^e) = \alpha_1\Delta\varepsilon_1^e + \alpha_2(\Delta\varepsilon_2^e + \Delta\varepsilon_3^e)\\ \Delta\sigma_2 &= S_2(\Delta\varepsilon_1^e, \Delta\varepsilon_2^e, \Delta\varepsilon_3^e) = \alpha_1\Delta\varepsilon_2^e + \alpha_2(\Delta\varepsilon_1^e + \Delta\varepsilon_3^e)\\ \Delta\sigma_3 &= S_3(\Delta\varepsilon_1^e, \Delta\varepsilon_2^e, \Delta\varepsilon_3^e) = \alpha_1\Delta\varepsilon_3^e + \alpha_2(\Delta\varepsilon_1^e + \Delta\varepsilon_2^e)\end{aligned} \tag{3.96}$$

Where: constant α_2 is given by equation (3.60) and $\alpha_1 = K + \frac{4}{3}G$ (3.97)

Plastic potential of the non-associated flow rule for shear failure g^s is given by:

$$g^s = \sigma_1 - \sigma_3 N_\psi \tag{3.98}$$

Where ψ is the dilation angle and constant N_ψ is given by:

$$N_\psi = \frac{1+\sin\psi}{1-\sin\psi} \tag{3.99}$$

Plastic potential of the associated flow rule for tension failure g^t is given by:

$$g^t = \sigma_3 \tag{3.100}$$

Partial differentiations of equation (3.98), required to obtain plastic correction for shear failure, are given by:

$$\frac{\partial g^s}{\partial\sigma_1} = 1;\ \frac{\partial g^s}{\partial\sigma_2} = 0\ ;\ \frac{\partial g^s}{\partial\sigma_3} = -N_\psi \tag{3.101}$$

If $\Delta\varepsilon_1^e$, $\Delta\varepsilon_2^e$ and $\Delta\varepsilon_3^e$ are substituted by the above differentiations in equation (3.96) we get:

$$S_1\left(\frac{\partial g^s}{\partial \sigma_1}, \frac{\partial g^s}{\partial \sigma_2}, \frac{\partial g^s}{\partial \sigma_3}\right) = \alpha_1 - \alpha_2 N_\psi$$
$$S_2\left(\frac{\partial g^s}{\partial \sigma_1}, \frac{\partial g^s}{\partial \sigma_2}, \frac{\partial g^s}{\partial \sigma_3}\right) = \alpha_2 (1 - N_\psi) \qquad (3.102)$$
$$S_3\left(\frac{\partial g^s}{\partial \sigma_1}, \frac{\partial g^s}{\partial \sigma_2}, \frac{\partial g^s}{\partial \sigma_3}\right) = -\alpha_1 N_\psi + \alpha_2$$

We get from equations (3.57), (3.90) and (3.102) by having $f = f^s$ the following:

$$\sigma_1^N = \sigma_1^I - \lambda^s (\alpha_1 - \alpha_2 N_\psi)$$
$$\sigma_2^N = \sigma_2^I - \lambda^s \alpha_2 (1 - N_\psi) \qquad (3.103)$$
$$\sigma_3^N = \sigma_3^I - \lambda^s (-\alpha_1 N_\psi + \alpha_2)$$

Where λ^s by using equation (3.56) is given by:

$$\lambda^s = \frac{f^s(\sigma_1^I, \sigma_3^I)}{(\alpha_1 - \alpha_2 N_\psi) - (-\alpha_1 N_\psi + \alpha_2) N_\phi} \qquad (3.104)$$

Partial differentiations of equation (3.100) for tensile failure are:

$$\frac{\partial g^t}{\partial \sigma_1} = 0\,;\ \frac{\partial g^t}{\partial \sigma_2} = 0\ ;\ \frac{\partial g^t}{\partial \sigma_3} = 1 \qquad (3.105)$$

If $\Delta\varepsilon_1^e$, $\Delta\varepsilon_2^e$ and $\Delta\varepsilon_3^e$ are substituted by the above differentiations in equation (3.96) gives:

$$S_1\left(\frac{\partial g^t}{\partial \sigma_1}, \frac{\partial g^t}{\partial \sigma_2}, \frac{\partial g^t}{\partial \sigma_3}\right) = \alpha_2 \qquad S_3\left(\frac{\partial g^t}{\partial \sigma_1}, \frac{\partial g^t}{\partial \sigma_2}, \frac{\partial g^t}{\partial \sigma_3}\right) = \alpha_1 \qquad (3.106)$$
$$S_2\left(\frac{\partial g^t}{\partial \sigma_1}, \frac{\partial g^t}{\partial \sigma_2}, \frac{\partial g^t}{\partial \sigma_3}\right) = \alpha_2$$

By having $f = f^t$ from equations (3.56), (3.57), we get the following:

$$\sigma_1^N = \sigma_1^I - (\sigma_3^I - \sigma^t)\frac{\alpha_2}{\alpha_1}$$

$$\sigma_2^N = \sigma_2^I - (\sigma_3^I - \sigma^t)\frac{\alpha_2}{\alpha_1} \quad ; \quad \lambda^t = \frac{\sigma_3^I - \sigma^t}{\alpha_1} \tag{3.107}$$

$$\sigma_3^N = \sigma^t$$

New principal stresses are computed following similar computations algorithm in this model to the Drucker-Prager algorithm explained before.

Five other constitutive models in plastic model group are not detailed here and will not be used in this thesis. These five models are: Ubiquitous-Joint Model, Bilinear Strain-Hardening/Softening Ubiquitous-Joint Model, Double-Yield Model, Modified Cam-Clay Model and Hoek-Brown Model.

3.7 Rationale for using the Flac^{3D} Program

3.7.1 Applications [30]

Flac^{3D} has many capabilities and features making it one of the most important programs that can solve and research great numbers of problems in Geo mechanics. Examples of these problems are simulating sequential excavation in tunnel and mine design, loading capacity and deformations in slope and foundation design, cable support on geological materials, fully saturated fluid flow, time-dependant behaviour of viscous materials, dynamic effect of explosive loading and vibration, and many other problems.

3.7.2 Advantages [31]

Although Flac^{3D} transforms, similar to the finite element method, the differential equilibrium equations into elemental matrices relate nodal forces with nodal displacements (velocities), as shown before, there is no need to build up the global stiffness matrix which saves computer memory and shortens the runtime when solving three dimensional problems. Non-linear stress-

strain problems are solved as fast as the linear problems by Flac$^{3D.}$ The mixed discretization technique explained before increases the accuracy of solving plastic flow problems.

3.7.3 FISH Programming Language [33]

FISH is a programming language embedded in Flac^{3D} giving the user the capabilities to define new variables and functions. It helps the user to plot and print new variables, generate a special type of grid not available readily in Flac^{3D} shapes library, and specify unusual distributions of properties that cannot be assigned by the gradient command or other commands, and it helps to execute coupling BEM method with Flac^{3D} as the main objective of this thesis.

FISH program is written in the Flac^{3D} data file as a normal part of it. Statements between the words *define* and *end* are processed as a FISH function. Functions may invoke other functions, which may invoke others, and so on. FISH has intrinsic functions (such as *cos, sqrt, log* …) and statements (such as *loop* and *end loop, if-else* and *end if*…) similar to those in other programming languages (FORTRAN, C^{++}…). Users can also create their own intrinsic functions by using the C++ FISH intrinsic plug-in feature. There are a great number of FISH variables that allow the user to extract a value or assign a value to terms in Flac^{3D} grid points (nodes), zones, interfaces and structural elements. Examples on grid point variables are:

1-*gp-xdisp* (*p-gp*) is x-displacement at grid point.

Where *p-gp = gp-near* (*x, y, z*) is the address of the grid point closest to (x, y, z).

2-*gp-yfapp* (*p-gp*) is y-applied force including body force at grid point.

3-*gp-zfunbal* (*p-gp*) is z-unbalanced force at grid point.

4-*gp-xvel* (*p-gp*) is x-velocity at grid point.

Examples on zone variables are:

1- *z-ssr* (*p-z*) is zone shear strain rate.

Where *p-z = z-near* (*x*, *y*, *z*) is the address of the zone closest to (x, y, z).

2- *z-sig1* (*p-z*) is zone major principal stress.

3- *z-sxx* (*p-z*) is zone xx-stress.

4- *z-syz* (*p-z*) is zone yz-stress.

There are so many FISH variables that can be seen listed in the Flac^{3D} manual. These variables are very essential to compute boundary tractions (stresses) and displacements from the Flac^{3D} program and read them as an input data for the BEM code embedded in the Flac^{3D} data file.

Chapter 4

Literature Review

Coupling the BEM with other Numerical Methods

4.1 Introduction

Commercial programs are used in the existing research works for different purposes such as validating the developed coupled FE or FD with BE methods [116] [121], combining the advantages of a commercial FEM program with a developed BEM code [42], [122], verifying a developed commercial FEM program capabilities of solving a coupled problems with complicated geometry [123] and other applications. Cermak et al. [45], in 1968, and Zienkiewicz et al. [46] in 1977, pioneered the research to couple FDM with BEM, and FEM with BEM, respectively.

4.2 Coupling BEM with FEM

Since 1977, coupling BEM with FEM has been investigated extensively and applied to different areas such as fluid mechanics, fracture mechanics, and geo-mechanics. A detailed literature review in coupling FEM with BEM can be found in many references [41], [43], [44], [47-51] . Beer and Meek [56], Wendland [52], Mitsui et al. [53], Varadarajan et al. [54] , Swoboda et al. [55], Feng and Owen [57], Estrof and Firuziaan [58], Chen and Zhao[59] and Rizos and Wang [60] used coupling BEM with FEM applications in the geotechnical field. The scientists, beginning with Li et al. [47] and followed by Lin et al. [51], Ganguly [48], and Elliethy et al. [41] classified the existing BEM/FEM coupling approaches into three groups: first, boundary element method approach; second, finite element method approach; and third, those not belonging to either of the previous two groups. The first two approaches achieved the

coupling at the level of the discretized equations, while the third approach developed an iterative domain decomposition coupling method.

4.2.1 Coupling at the Level of Discretized Equations

The discretized equations for BEM and FEM sub-domains in both FE and BE approaches are combined to build an entire unified system of equations for the whole domain [41]. The research based on the first approach [3], [46], [47], [56], [61], [62], [63] treats the BEM sub-domain or sub-domains as finite elements or super finite elements, and assembles its or their pseudo-stiffness matrices with the FE sub-domains global stiffness matrix. The BEM sub-domains' traction-displacement relations, transformed into force-displacement relations, are assembled with the global force-displacement relation of FEM sub-domains [47]. The disadvantage of the FEM approach is that combining the asymmetric and fully populated matrix of BEM with the symmetric and sparsely-banded stiffness matrix of FEM produces an asymmetric assembled system of equations [41], [51], which costs more to solve on the computer than the symmetric system[57].

In the BEM based approach [3], [61], [64], [65],the force-displacement relations, transformed into traction-displacement relations, of the FEM sub-domains, considered as equivalent boundary element regions, are assembled with the global traction-displacement relations and then solved [47].The disadvantages of the BEM approach is that the method destroys the symmetry and bandedness, positive characteristics the FEM originally possesses [51].

Brebbia and Georgiou [61], for example, presented the FEM approach, based on Zienkiewicz et al. [46] work, and the BEM approach for 2D elastostatics problems as the following:

The domain in figure 4.1a consists of two domains Ω^1 and Ω^2 joined by an interface r^I. The domain Ω^1 is assumed to be analyzed numerically using BEM and expressed in matrix form as:

$$\mathbf{HU} = \mathbf{GP} + \mathbf{B} \tag{4.1}$$

Where:

H and **G** are global matrices (influence factors).

U, **P** and **B** are boundary nodal displacement, traction, and body force global vectors, respectively.

The domain Ω^2 is analyzed using FEM and expressed in matrix form as:

$$\mathbf{KU} = \mathbf{F} + \mathbf{D} \tag{4.2}$$

Where:

K is the stiffness matrix, **F** is the equivalent nodal force global vector, **U** nodal displacement global vector, and **D** body force global vector.

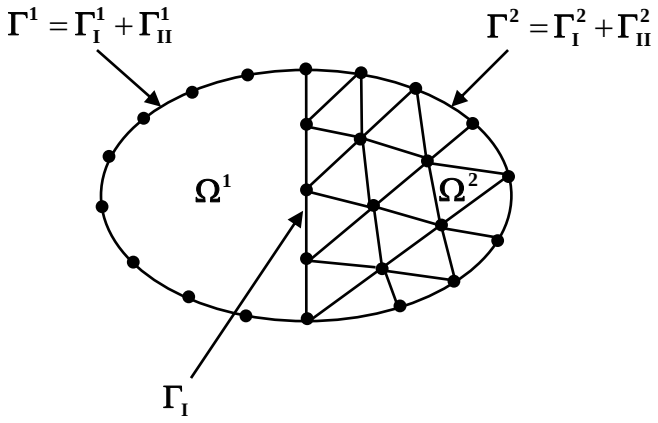


Figure 4.1a. Body divided into finite elements and boundary elements [61]

Because both systems in equations 4.1 and 4.2 are expressed in terms of two different variables, the nodal traction vector **P** can be transformed into equivalent nodal force vector by:

$$\mathbf{F} = \mathbf{MP} \tag{4.3}$$

Where:

M is a matrix due to weighing of the boundary tractions by the interpolation functions for the displacement and can be found from:

$$\mathbf{U}^{*}\mathbf{M}\mathbf{P} = \Sigma\left[\mathbf{u}^{*\,n,T}\{\int_{\Gamma_l}\Phi\,\Psi^{T}.d\Gamma\}\mathbf{p}_k^{n}\right] \tag{4.4}$$

Where:

$\mathbf{U}^{*}$, $\mathbf{u}^{*nT}$ are the global vector of virtual displacement. Φ, Ψ are displacement and traction interpolation functions, respectively. $\mathbf{p}_k^n$ is nodal traction vector, while Γ_l the boundary element surface.

If equation (4.3) is substituted in equation (4.2), it will take the form:

$$\mathbf{K}\mathbf{U} = \mathbf{M}\mathbf{P} + \mathbf{D} \tag{4.5}$$

Compatibility and equilibrium conditions at the interface Γ^{I} are:

$$\mathbf{U}_{\mathrm{I}}^{1} = \mathbf{U}_{\mathrm{I}}^{2} \tag{4.6}$$

$$\mathbf{P}_{\mathrm{I}}^{1} + \mathbf{P}_{\mathrm{I}}^{2} = \mathbf{0} \tag{4.7}$$

Where:

$\mathbf{U}_{\mathrm{I}}^{1}, \mathbf{U}_{\mathrm{I}}^{2}$ are the displacement vectors at the interface Γ^{I} for both regions 1 and 2, respectively. $\mathbf{P}_{\mathrm{I}}^{1}, \mathbf{P}_{\mathrm{I}}^{2}$ are the traction vectors at the interface Γ^{I} for both regions 1 and 2, respectively.

The FEM approach considers the boundary element region Ω^1 as an equivalent finite element and it incorporates its effective stiffness matrix into the global stiffness system of region Ω^2 as follows:

After inverting matrix G, equation (4.1) becomes:

$$\mathbf{K}'\mathbf{U} = \mathbf{F}' + \mathbf{D}' \tag{4.8}$$

Where:

$$\mathbf{K'} = \mathbf{M}\mathbf{G}^{-1}\mathbf{H}$$
$$\mathbf{D'} = \mathbf{M}\mathbf{G}^{-1}\mathbf{B} \qquad (4.9)$$
$$\mathbf{F'} = \mathbf{M}\mathbf{P}$$

Matrix **K'** is asymmetric. The system in equation (4.8) may be assembled with the system in equation (4.2) of region 2 to form a global system, to be solved, but only after symmetrising **K'** and applying the compatibility and equilibrium conditions of equations (4.6) and (4.7). Many researchers worked on symmetrising the asymmetric matrix **K'**. Zienkiwicz et al. [46] achieved the symmetry based on the energy-variation concept. Brebbia and Georgiou [61] symmetrized it by minimizing the square of the errors in non-symmetric off-diagonal terms. Although Kohno et al. [122] and Swoboda et al. [55] confirmed the success of the symmetrisation, Tullberg and Bolteus [62], Mang et al. [66] and Beer et al. [3] reported a loss of accuracy in the area close to the interface. The disadvantage of the above explained coupling technique, known as a global coupling approach, is the necessity to invert the coefficient matrix **G** for the whole boundary element region Ω^1, which consumes expensive computer time [68]. Conversely, in single condensation developed by Beer [3], [67] and bi-condensation developed by Li et al. [47] (both techniques are called local coupling approach), all degrees of freedom for the non-interfacing nodes were condensed, which restrict the inversion of BEM-related coefficient matrices to a condensed matrix referred to the interface.

Brebbia and Georgiou in the BEM approach considered region 2 as a boundary element type which allowed them to combine both systems in equations (4.1) and (4.5) as the following:

Equation (4.1) of region 1 and equation (4.5) of region 2 may be rewritten as:

$$\left[\mathbf{H}^1 \mathbf{H}_\mathbf{I}^1\right]\begin{Bmatrix}\mathbf{U}^1\\ \mathbf{U}_\mathbf{I}^1\end{Bmatrix} = \left[\mathbf{G}^1 \mathbf{G}_\mathbf{I}^1\right]\begin{Bmatrix}\mathbf{P}^1\\ \mathbf{P}_\mathbf{I}^1\end{Bmatrix} + \mathbf{B}^1 \qquad (4.10)$$

$$\left[\mathbf{K}^2\ \mathbf{K}_\mathbf{I}^2\right]\begin{Bmatrix}\mathbf{U}^2\\ \mathbf{U}_\mathbf{I}^2\end{Bmatrix}=\left[\mathbf{M}^2\ \mathbf{M}_\mathbf{I}^2\right]\begin{Bmatrix}\mathbf{P}^2\\ \mathbf{P}_\mathbf{I}^2\end{Bmatrix}+\mathbf{D}^2 \tag{4.11}$$

The boundary conditions in equations (4.6) and (4.7) can be rewritten as:

$$\mathbf{U}_\mathbf{I}=\mathbf{U}_\mathbf{I}^1=\mathbf{U}_\mathbf{I}^2 \tag{4.12}$$

$$\mathbf{P}_\mathbf{I}=\mathbf{P}_\mathbf{I}^1=-\mathbf{P}_\mathbf{I}^2 \tag{4.13}$$

By substituting equations (4.12) and (4.13) in equations (4.10) and (4.11), and by combining them, we will get the assembled system:

$$\begin{bmatrix}\mathbf{H}^1 & \mathbf{H}_\mathbf{I}^1 & -\mathbf{G}_\mathbf{I}^1 & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_\mathbf{I}^2 & \mathbf{M}_\mathbf{I}^2 & \mathbf{K}^2\end{bmatrix}\begin{Bmatrix}\mathbf{U}^1\\ \mathbf{U}_\mathbf{I}\\ \mathbf{P}_\mathbf{I}\\ \mathbf{U}^2\end{Bmatrix}=\begin{bmatrix}\mathbf{G}^1 & \mathbf{0}\\ \mathbf{0} & \mathbf{M}^2\end{bmatrix}\begin{Bmatrix}\mathbf{P}^1\\ \mathbf{P}^2\end{Bmatrix}+\begin{Bmatrix}\mathbf{B}^1\\ \mathbf{D}^1\end{Bmatrix} \tag{4.14}$$

The system is ready for solving to obtain the unknowns of both regions. This approach does not require a matrix inversion.

Beer [3], [67], as a second coupling example, developed a local coupling of FEM with BEM as follows:

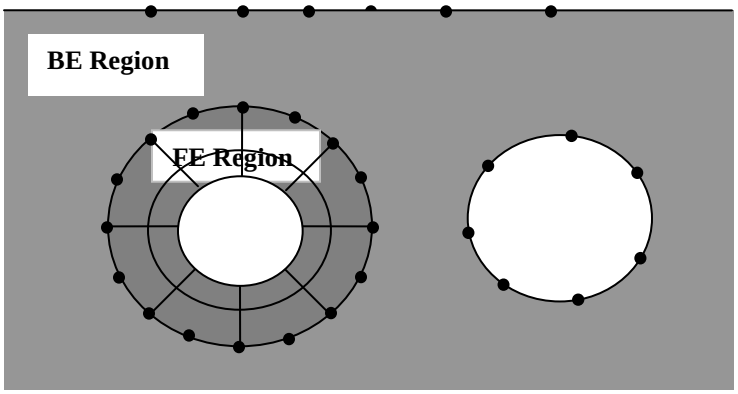


Figure 4.1b. Partially Coupled BE and FE Example [3]

Two partially coupled regions (only part of the BE region is coupled with the FE region) are shown in Figure 4.1b. In the FEM based approach the nodal traction vector $\{\mathbf{t}\}_c$ at the interface in the BE region is given by:

$$\{\mathbf{t}\}_c = \{\mathbf{t}\}_{c0} + \mathbf{K}_{BE}\{\mathbf{u}\}_c \tag{4.15}$$

Where:

$\{\mathbf{u}\}_c$ is the nodal displacement vector at the interface.

$\mathbf{K}_{BE}$ is the pseudo stiffness matrix, obtained using equation (4.1), by applying zero prescribed Dirichlet or Neumann BCs ($\{\mathbf{u}\}_f = 0$ or $\{\mathbf{t}\}_f = 0$) on the non-interfacing nodes of the BE region and unit displacement at every node i of the interface in turn. Computed traction at the interface $\{\mathbf{t}\}_{ci}$ is placed in column i of matrix $\mathbf{K}_{BE}$. The Matrix $\mathbf{K}_{BE}$ columns' total number equals the interface degrees of freedom total number.

$\{\mathbf{t}\}_{c0}$ is the traction vector at the interface. This vector is obtained from equation (4.1) by applying Dirichlet boundary conditions ($\{\mathbf{u}\}_c = 0$) on the interface nodes and by applying the prescribed boundary values on the non-interfacing nodes of the BE region.

The relationship between nodal force $\{\mathbf{F}\}_c$ and nodal displacement $\{\mathbf{u}\}_c$ vectors at the interface in the FE region is:

$$\{\mathbf{F}\}_c = \{\mathbf{F}\}_{c0} + \mathbf{K}_{FE}\{\mathbf{u}\}_c \tag{4.16}$$

Where:

$\{\mathbf{F}\}_{c0}$ is the interface nodal force vector computed using equation (4.2) after fixing the interface nodes ($\{\mathbf{u}\}_c = \mathbf{0}$). $\mathbf{K}_{FE}$ is the condensed stiffness matrix of the FE region that involves only the interface nodes.

The tractions at the BE region interface should be transformed into equivalent nodal forces using the principal of virtual work before coupling and making equation (4.15) have the form:

$$\{\mathbf{F}\}_c = \mathbf{N}\{\mathbf{t}\}_c = \mathbf{N}\{\mathbf{t}\}_{c0} + \mathbf{N}\mathbf{K}_{\mathbf{BE}}\{\mathbf{u}\}_c \tag{4.17}$$

Where:

N is the interface converting matrix, very similar to matrix **M** in equation (4.4) that interpolates the traction over interface boundary elements. $\mathbf{NK}_{\mathbf{BE}}$ is the true stiffness matrix of the BE region interface. This matrix is asymmetric since $\mathbf{K}_{\mathbf{BE}}$ is asymmetric too. The matrix $\mathbf{NK}_{\mathbf{BE}}$ should be assembled into a global stiffness matrix in equation (4.2) treating the BE region as a super finite element, but only after symmetrising the true stiffness matrix and applying the compatibility and equilibrium conditions of equations (4.6) and (4.7), to have the problem solved and the unknown variables computed.

In the BEM based approach the finite elements will be treated as the BE region has its own stiffness matrix after reversing the above procedure, equation (4.16) and (4.17) become:

$$\{\mathbf{t}\}_c = \mathbf{N}^{-1}\{\mathbf{F}\}_c = \mathbf{N}^{-1}\{\mathbf{F}\}_{c0} + \mathbf{N}^{-1}\mathbf{K}_{\mathbf{FE}}\{\mathbf{u}\}_c \tag{4.18}$$

$$\{\mathbf{t}\}_c = \{\mathbf{t}\}_{c0} + \mathbf{K}_{\mathbf{BE}}\{\mathbf{u}\}_c \tag{4.19}$$

4.2.2 Iterative Coupling of BEM with FEM

In addition to the disadvantages of BEM and FEM approaches mentioned earlier, these coupling methods require building a complicated unified system of equations, contrary to the uncoupled BE or FE, which requires assembling a simple separate system of equations for each single method [41]. The available FEM and BEM programs are built very differently in data structures, program organization and numerical techniques [57]. Consequently, constructing an integrated FE/BE software environment needs substantial effort and demands more computer processing time for complicated geometry. Moreover, many existing commercial software, such

as Flac^{3D}, do not build stiffness matrix to be coupled with BE regions pseudo-stiffness matrices, as the FLAC^{3D} manual states: " it is not necessary to store any matrices, which means: (a) a large number of elements may be modeled with a modest memory requirement; and (b) a large-strain simulation is hardly more time-consuming than a small-strain run because there is no stiffness matrix to be updated" [31]. Cruse et al. [69] concluded from the experience with current coupling procedures that it is necessary to preserve the nature of the BEM, rather than to force it into a finite element format [57].

Iterative domain decomposition methods [51], [57], [68], [70-90] were developed relatively recently to evade the disadvantages of coupling at the level of discretized equations. The main advantages of these methods are:

1. There is no need to combine matrices of the BEM sub-domains with the FEM sub-domain. Instead, separate computing is performed for each sub-domain and by successive renewal of the variables on the interface, the solution convergence is attained [41] and the continuity and equilibrium conditions at the interface are satisfied.
2. Problems analyzed using coupling BEM and FEM, and treated as sub-problems, with different formulations require little or no modifications of the existing computer codes [51].
3. Iterative coupling of available or even not explicitly available BEM and FEM computer source codes, with complex constitutive laws and reduced computer cost, can be implemented interactively [51].

Domain Decomposition Methods (DDM):

DDM partitions the task of solving partial differential equations (PDE) numerically by splitting the original problem of a large and/or complex domain into a set of sub-domains [70]. The resulting multi-PDE system can be coupled using relaxation operators on the

interfacing boundaries. These methods may be differentiated into two groups: non-overlapping and overlapping Schwarz domain decomposition methods. In the overlapping Schwarz method the interfacing sub-domains overlap a common region; references [91-92] provide examples of utilizing the overlapping Schwarz domain decomposition methods in coupling BEM with FEM. Although both methods have proved to be an effective numerical procedure and enjoy accurate good convergence properties, recent studies revealed that a non-overlapping method can relieve the user from certain complications and compete better than the overlapping method in its formulation and implementation [71]. The non-overlapping method can be viewed from the preconditioning and the interface relaxation point views. Rice et al. [71] developed two interface relaxation methods and presented a few others. In these methods the following main steps are repeated:

1. The domain is partitioned into non-overlapping sub-domains and the boundary conditions on the interface boundaries are imposed.

2. Initial guesses on the interfaces are assumed then the set of the resulting PDE problems are solved.

3. If the obtained solutions do not satisfy the interface boundary conditions, the interface relaxation is applied to obtain new interface boundary values. The PDEs with these new values are solved.

4. The above steps are repeated until convergence.

Let us have the global differential problem in the domain Ω to be expressed locally in the sub-domain Ω_i by the following system of loosely coupled differential problems [71]:

$$D_i u = f_i \text{ in } \Omega_i \text{ for } i=1, 2, ...p$$

$$G_i\, u = 0 \quad \text{on} \quad \partial\Omega_i \backslash \partial\Omega \tag{4.20}$$

$$B_i\, u = c_i \quad \text{on} \quad \partial\Omega_i \cap \partial\Omega$$

Where:

D_i is a non-linear differential operator. p is the number of sub-domains. G_i is a local interface differential operator. $\partial\Omega_i$, $\partial\Omega$ are local and global boundary of domain Ω and sub-domain Ω_i respectively. B_i is a local boundary condition operator. PDEs are coupled through the interface conditions $G_i\, u = 0$. c_i is a continuous function.

The following relaxation methods developed or presented by Rice et al. [71] are some of many existing relaxation methods based on the general mathematical definition in equation (4.20):

1. The Dirichlet/Neumann averaging method algorithm:

for k = 0,1,2....

$u^{(k+1/2)}$ = solve_pde(ui) in each sub-domain,

$$\text{dui} = \beta \frac{\partial u_R^{(k+1/2)}}{\partial x} + (1-\beta) \frac{\partial u_L^{(k+1/2)}}{\partial x} \quad \text{on each interface,} \tag{4.21}$$

$u^{(k+1)}$ = solve_pde(dui) in each sub-domain,

$\text{ui} = \alpha u_R^{(k+1)} + (1-\alpha) u_L^{(k+1)}$ on each interface.

Where:

ui, dui is the solution and its gradient of the problem at the interface associated with sub-domain Ω_i. u = solve pde(ui , dui) obtains the solution u of the local PDE problem with Dirichlet/Neumann boundary condition on the interface using the interface values ui and dui , respectively. R and L denote left and right sub-domains or interface respectively. α and β are relaxation parameters.

This method is investigated further in references [72-75].

2. The geometric contraction based method algorithm:

for k = 0,1,2.....

$$ui^{(k+1)} = ui^{(k)} - \alpha\left(\frac{\partial u_L^{(k)}}{\partial x} - \frac{\partial u_R^{(k)}}{\partial x}\right) \text{ on each interface,} \tag{4.22}$$

$u^{(k+1)}$ = solve_pde($dui^{(k+1)}$) in each sub-domain.

Where: α is a relaxation parameter.

3. The Robin relaxation method algorithm:

for k = 0,1,2...

Solve equations (4.20) in each sub-domain with:

$$-\frac{\partial u^{(k+1)}}{\partial x} + \rho u^{(k+1)} = -\frac{\partial u_L^{(k)}}{\partial x} + \rho u_L^{(k)} \text{ on sub-domain's left interface,} \tag{4.23}$$

$$-\frac{\partial u^{(k+1)}}{\partial x} + \rho u^{(k+1)} = -\frac{\partial u_R^{(k)}}{\partial x} + \rho u_R^{(k)} \text{ on sub-domain's right interface,}$$

Where: ρ is a relaxation parameter.

The method was developed first by Lions [76].

4. The Steklov-Poincaré's operator method algorithm:

for k = 0,1,2...

$u^{(k+1/2)}$ = solve_pde(ui) in each sub-domain,

$$dui = \frac{1}{2}\left(\frac{\partial u_L^{(k+1/2)}}{\partial x} + \frac{\partial u_R^{(k+1/2)}}{\partial x}\right) \text{ on each interface,} \tag{4.24}$$

$u^{(k+1)}$ = solve_pde0(dui) (Lu=0)in each sub-domain,

$$ui = ui - \frac{1}{2}\rho\left(u_R^{(k+1)} + u_L^{(k+1)}\right) \text{ on each interface.}$$

This method was analysed from the preconditioning viewpoint only and not from the interface relaxation viewpoint by Tallec et Al [77].

Perera et al. [80], [81] applied the Steklov–Poincaré operator method to solve Laplace [80] and elasto-plasticity [81] problems using non-overlapping iterative domain decomposition FEM-BEM coupling method. The Steklov-Poincaré operator method consists of two parallel Dirichlet-Nuemann coupling steps, as shown in the method algorithm earlier. In the first step the compatibility condition is applied by assuming the same Dirichlet condition on the interface between the FEM and BEM sub-domains and a parallel Dirichlet problem for both sub-domains is solved. In the second step the equilibrium condition in the interface is introduced using the Steklov–Poincaré operator (or called Schur complement), which represents the nodal loads in the interface (approached from the FEM sub-domain) for a problem of Dirichlet conditions applied on the external boundary, and a parallel Nuemann problem for the FEM and BEM sub-domains is solved [80].

Elleithy and Tanaka [70], [78] proposed to apply the geometric contraction based on Robin relaxation and Dirichlet/Neumann averaging methods developed by Rice et al. [71] , Lions [76] and Mu et al. [72], respectively to solve elasto-plastic, and Laplace problems using non overlapping iterative domain decomposition FEM-BEM coupling method as the following [78]

Domain Ω, in Figure 4.2, of the global differential problem is decomposed into a finite element sub-domain Ω_F governed by the following potential (displacement)-flux (force) equation:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_F^F \\ \mathbf{u}_F^I \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_F^F \\ \mathbf{f}_F^I \end{Bmatrix} \tag{4.25}$$

and a boundary element sub-domain Ω_B represented by the integral equation:

$$\begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_B^B \\ \mathbf{u}_B^I \end{Bmatrix} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{t}_B^B \\ \mathbf{t}_B^I \end{Bmatrix} \tag{4.26}$$

Where:

$\mathbf{u}_F^I$ and $\mathbf{f}_F^I$ are interface potential (displacement) and integrated flux (force), respectively approached from the FEM sub-domain.

$\mathbf{u}_F^F$ and $\mathbf{f}_F^F$ are non-interface potential (displacement) and integrated flux (force), respectively in the FEM sub-domain.

$\mathbf{K}_{11}$, $\mathbf{K}_{12}$, $\mathbf{K}_{22}$ and $\mathbf{K}_{21}$ are stiffness matrices.

$\mathbf{u}_B^I$ and $\mathbf{t}_B^I$ are interface potential (displacement) and flux (traction), respectively approached from the BEM sub-domain.

$\mathbf{u}_B^B$ and $\mathbf{t}_B^B$ are non-interface potential (displacement) and flux (traction), respectively in the BEM sub-domain.

$\mathbf{H}_{11}$, $\mathbf{H}_{12}$, $\mathbf{H}_{22}$, $\mathbf{H}_{21}$, $\mathbf{K}_{11}$, $\mathbf{K}_{12}$, $\mathbf{K}_{22}$ and $\mathbf{K}_{21}$ are influence coefficient matrices.

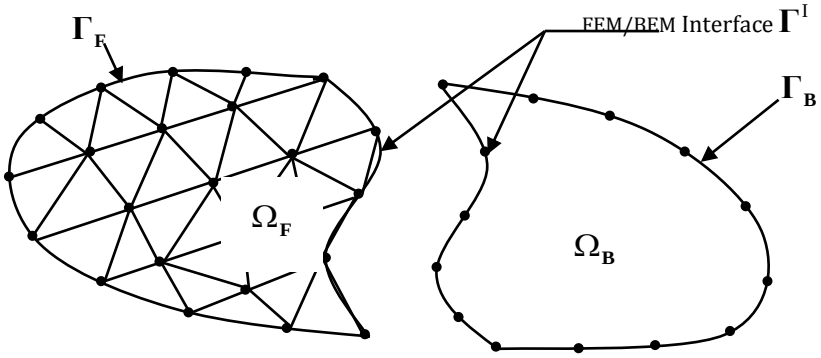


Figure 4.2 Domain decomposition into BEM and FEM sub-domains [78]

- The geometric contraction based FEM–BEM coupling algorithm:

For n =0, 1, 2, ... continue until convergence,

1. Set initial guess for $\{\mathbf{u}_{F,0}^I\}$ and $\{\mathbf{u}_{B,0}^I\}$,

2. Obtain $\{\mathbf{t}^{I}_{F,n}\}$ in the FE sub-domain by solving equation (4.25) with the equation:

$$\{\mathbf{f}^{I}_{F,n}\} = [\mathbf{M}]\{\mathbf{t}^{I}_{F,n}\} \tag{4.27}$$

3. Obtain $\{\mathbf{t}^{I}_{B,n}\}$ in the BE sub-domain by solving equation (4.26),

4. Apply:

$$\{\mathbf{u}^{I}_{B,n+1}\} = \{\mathbf{u}^{I}_{B,n}\} - \alpha(\{\mathbf{t}^{I}_{B,n}\} + \{\mathbf{t}^{I}_{F,n}\}) \tag{4.28}$$

$$\{\mathbf{u}^{I}_{F,n+1}\} = \{\mathbf{u}^{I}_{B,n+1}\} \tag{4.29}$$

Where: α is a relaxation parameter.

5. Return to step (2) and repeat the steps that follow until convergence.

The new values of the potentials/displacements are computed in the above algorithm by adding to the old values, a combination of the normal boundary derivatives (fluxes/tractions) of the adjacent sub-domains weighted geometrically.

- Robin relaxation FEM-BEM coupling algorithm:

Define $\{\mathbf{c}^{I}_{B,n+1}\} = -\{\mathbf{t}^{I}_{F,n}\} + \rho\{\mathbf{u}^{I}_{F,n}\}$ and $\{\mathbf{c}^{I}_{F,n+1}\} = -\{\mathbf{t}^{I}_{B,n}\} + \rho\{\mathbf{u}^{I}_{B,n}\}$ (4.30)

Where: ρ is a relaxation parameter.

For n =0, 1, 2,... continue until convergence,

1. Set initial guess for $\{\mathbf{u}^{I}_{F,0}\}$ and $\{\mathbf{u}^{I}_{B,0}\}$,

2. Obtain $\{\mathbf{u}^{I}_{F,n}\}$ and $\{\mathbf{t}^{I}_{F,n}\}$ in the FE sub-domain by solving equation (4.25) with the equation (4.27),

3. Obtain $\{\mathbf{u}^{I}_{B,n}\}$ and $\{\mathbf{t}^{I}_{B,n}\}$ in the BE sub-domain by solving equation (4.26),

4. Apply:

$$\{\mathbf{c}^{I}_{B,n+1}\} = -\{\mathbf{t}^{I}_{F,n}\} + \rho\{\mathbf{u}^{I}_{F,n}\}$$

$$\{\mathbf{c}^{I}_{F,n+1}\} = -\{\mathbf{t}^{I}_{B,n}\} + \rho\{\mathbf{u}^{I}_{B,n}\}$$

5. Return to step (2) and repeat the steps that follow until convergence.

The new values of the potentials/displacements and fluxes/tractions on the interface are obtained in this algorithm by matching a convex combination of Dirichlet and Neumann data from the interfacing FE and BE sub-domains.

- The Dirichlet/ Neumann averaging FEM-BEM coupling algorithm:

For n =0, 2, 4, ... continue until convergence,

1. Set initial guess for $\{\mathbf{u}^{I}_{F,0}\}$ and $\{\mathbf{u}^{I}_{B,0}\}$,

2. Obtain $\{\mathbf{t}^{I}_{F,n}\}$ in the FE sub-domain by solving equation (4.25) with the equation (4.27),

3. Obtain $\{\mathbf{t}^{I}_{B,n}\}$ in the BE sub-domain by solving equation (4.26),

4. Apply:

$$\{\mathbf{t}^{I}_{B,n+1}\} = (1-\varphi_1)\{\mathbf{t}^{I}_{B,n}\} - \varphi_1\{\mathbf{t}^{I}_{F,n}\} \text{ and } \{\mathbf{t}^{I}_{F,n+1}\} = (1-\varphi_1)\{\mathbf{t}^{I}_{F,n}\} - \varphi_1\{\mathbf{t}^{I}_{B,n}\} \quad (4.31)$$

Where: φ_1 is a relaxation parameter.

5. Obtain $\{\mathbf{u}^{I}_{F,n+1}\}$ in the FE sub-domain by solving equation (4.25) with the equation (4.27),

6. Obtain $\{\mathbf{u}^{I}_{B,n+1}\}$ in the BE sub-domain by solving equation (4.26),

7. Apply:

$$\{\mathbf{u}^{I}_{B,n+2}\} = (1-\varphi_2)\{\mathbf{u}^{I}_{B,n+1}\} + \varphi_2\{\mathbf{u}^{I}_{F,n+1}\} \text{ and}$$

$$\{\mathbf{u}^{I}_{F,n+2}\} = (1-\varphi_2)\{\mathbf{u}^{I}_{F,n+1}\} + \varphi_2\{\mathbf{u}^{I}_{B,n+1}\} \quad (4.31)$$

Where: φ_2 is a relaxation parameter.

8. Return to step (2) and repeat the steps that follow until convergence.

Bjorstad and Widlund's [79] developed an iterative loosely-coupled domain decomposition technique. They partitioned the PDE problem into finite element sub-problems that are solved by direct methods; meanwhile, the interaction across the interfaces is handled by a conjugate gradient method. Gerstle et al. [68] developed a non-overlapping domain decomposition FEM-BEM coupling method using an iterative conjugate gradient solver based on Bjorstad and Widlund's [79] techniques. The method algorithm is [68]:

1. Trial displacements ($\{\mathbf{u}^{I}_{F,1}\} = \{\mathbf{u}^{I}_{B,1}\}$) are applied at each iteration on the FEM and BEM sub-domains' interface.

2. The resulted nodal tractions are converted to nodal forces $\{\mathbf{f}^{I}_{B,i}\}$ .The resulted nodal forces $\{\mathbf{f}^{I}_{B,i}\}$ and $\{\mathbf{f}^{I}_{F,i}\}$ are both treated by the conjugate gradient solver. These forces are resulted from implementing the BEM and FEM in the single sub-domains, repectively. The unbalanced forces $\{\mathbf{f}^{I}_{unbalancedi}\} = \{\mathbf{f}^{I}_{F,i}\} + \{\mathbf{f}^{I}_{B,i}\}$ are used by the solver to predict the next iteration interface nodal displacement values.

3. The iterations stop whenever the unbalanced forces become sufficiently small and the solution converges.

The number of iterations (m) in the conjugate gradient method (CG) required to solve a symmetric positive definite linear set of equations is less than or equal to the number of equations (n). The Solver algorithm is [68]:

For i =1,2, ...n

1. Set initial displacement $\mathbf{u}^{(1)}$ and compute the unbalanced forces or the residual $\mathbf{r}^{(1)}$ at iteration i =1,

2. Set $\mathbf{p}^{(1)} = \mathbf{r}^{(1)}$,

3. Compute the vector $[\mathbf{K}]\mathbf{p}^{(i)}$ using the equation:

$$[\mathbf{K}]\mathbf{p}^{(i)} = \frac{\nabla F(\mathbf{u}^{(1)} + \sigma^{(i)}\mathbf{p}^{(i)}) - \nabla F(\mathbf{u}^{(1)})}{\sigma^{(i)}} \tag{4.32}$$

4. Compute the new values of the displacement residual and search direction vectors by:

$$\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + a^{(i)}\mathbf{p}^{(i)} \tag{4.33}$$

$$\mathbf{r}^{(i+1)} = \mathbf{r}^{(i)} - a^{(i)}[\mathbf{K}]\mathbf{p}^{(i)} \tag{4.34}$$

$$\mathbf{p}^{(i+1)} = \mathbf{r}^{(i)} + b^{(i)}\mathbf{p}^{(i)} \tag{4.35}$$

5. Repeat the above last three steps until the norm of $\mathbf{r}^{(i+1)}$ equals zero or less than a specified tolerance.

Where:

[**K**] is the positive definite coefficient matrix and $\mathbf{p}^{(i)}$ is [**K**]-conjugate search direction vector.

$a^{(i)}$, $b^{(i)}$ and $\sigma^{(i)}$ are scalars given by:

$$a^{(i)} = \mathbf{p}^{(i)T}\mathbf{r}^{(i)} / (\mathbf{p}^{(i)T}[\mathbf{K}]\mathbf{p}^{(i)}) \tag{4.36}$$

$$b^{(i)} = -\mathbf{r}^{(i+1)}[\mathbf{K}]\mathbf{p}^{(i)} / (\mathbf{p}^{(i)T}[\mathbf{K}]\mathbf{p}^{(i)}) \tag{4.37}$$

$$\sigma^{(i)} = \frac{\sigma}{\left|\mathbf{p}^{(i)}\right|} \tag{4.38}$$

σ is a specified small scalar value taken usually as 10^{-5}.

F is a scalar quadratic function given by the equation:

$$F(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T[\mathbf{K}]\mathbf{u} - \mathbf{f}^T\mathbf{u} + c \tag{4.39}$$

To avoid the use of the explicit formation of [K], the nominator in equation (4.32) is obtained by taking the difference quotient of the gradients $\nabla F(\mathbf{u}^{(1)} + \sigma^{(i)}\mathbf{p}^{(i)})$ and $\nabla F(\mathbf{u}^{(i)})$ according to equation (4.39).

Two parallel Neumann-Neumann and Dirichlet-Neumann non overlapping domain decomposition FEM-BEM iterative coupling methods were developed by Kamiya and Iwase [82], studied and outlined by Elleithy and Tanaka [78], to solve Laplace problems.

Boundary conditions (see Figure 4.2) for the boundary **r** are:

$$\left.\begin{array}{ll} \text{i.} & \{\mathbf{u}_B^B\} = \{\bar{\mathbf{u}}\} \text{ on } \mathbf{r}^B \\ \text{ii.} & \{\mathbf{t}_F^F\} = \{\bar{\mathbf{t}}\} \text{ on } \mathbf{r}^F \end{array}\right\} \tag{4.40}$$

Where: $\{\bar{\mathbf{u}}\}$ and $\{\bar{\mathbf{t}}\}$ are the prescribed boundary conditions.

- The parallel Neumann-Neumann algorithm is:

For n =0, 1, 2, ... continue until convergence,

1. Set initial guess for $\{\mathbf{t}_{B,0}^I\}$ and $\{\mathbf{t}_{F,0}^I\}$,

2. Obtain $\{\mathbf{u}^{I}_{F,n}\}$ in the FE sub-domain by solving equation (4.25) with equation (4.27),

3. Obtain $\{\mathbf{u}^{I}_{B,n}\}$ in the BE sub-domain by solving equation (4.26),

4. Apply:

$$\{\mathbf{t}^{I}_{B,n+1}\} = \{\mathbf{t}^{I}_{B,n}\} + \beta(\{\mathbf{u}^{I}_{F,n}\} - \{\mathbf{u}^{I}_{B,n}\}) \quad (4.41)$$

$$\{\mathbf{t}^{I}_{F,n+1}\} = -\{\mathbf{t}^{I}_{B,n+1}\} \quad (4.42)$$

Where: β is a relaxation parameter.

5. Return to step (2) and repeat the steps that follow until convergence.

- The parallel Dirichlet-Neumann algorithm is:

For n =0, 1, 2, ... continue until convergence,

1. Set initial guess for $\{\mathbf{u}^{I}_{B,0}\}$ and $\{\mathbf{t}^{I}_{F,0}\}$,

2. Obtain $\{\mathbf{t}^{I}_{B,n}\}$ in the BE sub-domain by solving equation (4.26),

3. Obtain $\{\mathbf{u}^{I}_{F,n}\}$ in the FE sub-domain by solving equation (4.25) with equation (4.27),

4. Apply:

$$\{\mathbf{u}^{I}_{B,n+1}\} = \{\mathbf{u}^{I}_{B,n}\} + \gamma(\{\mathbf{u}^{I}_{F,n}\} - \{\mathbf{u}^{I}_{B,n}\}) \quad (4.43)$$

$$\{\mathbf{t}^{I}_{F,n+1}\} = -\{\mathbf{t}^{I}_{B,n}\} \quad (4.44)$$

Where: γ is a relaxation parameter.

5. Return to step (2) and repeat the steps that follow until convergence.

Kamiya and Iwase [83] replaced the parallel Neumann-Neumann and Dirichlet-Neumann methods they used before with the conjugate gradient method, based on Glowinski et al. [84] work to renew the unknown integrated flux $\{\mathbf{f}^{\mathrm{I}}\}$ at the interface between the FEM and BEM sub-domains. This method has the advantage of determining the iteration renewal parameters automatically. Conversely, in the parallel methods the parameters are determined by the users depending on their own deep experience. Kamiya and Iwase [83] utilized the condense methods, introduced into the world of iterative coupling techniques and domain decomposition methods by Kane et al. [85], Navon and Cai [86], and, Doltsinis and Nolting [87]. In this method only the unknowns on the interface are treated during the renewal iteration. The condense method reduces equation (4.25) for the FEM sub-domain to the form:

$$[\mathbf{K}_{\mathbf{F}}^{*}]\left\{\mathbf{u}_{\mathrm{F}}^{\mathrm{I}}\right\}=\{\mathbf{f}_{\mathrm{F}}^{*}\} \tag{4.45}$$

Where:

$$[\mathbf{K}_{\mathbf{F}}^{*}]=[\mathbf{K}_{22}]-[\mathbf{K}_{21}][\mathbf{K}_{11}^{-1}][\mathbf{K}_{12}] \tag{4.46}$$

$$\{\mathbf{f}_{\mathbf{F}}^{*}\}=\{\mathbf{f}_{\mathrm{F}}^{\mathrm{I}}\}-[\mathbf{K}_{21}][\mathbf{K}_{11}^{-1}]\{\mathbf{f}_{\mathrm{F}}^{\mathrm{F}}\} \tag{4.47}$$

Equation (4.26) is rewritten first as the following:

$$\begin{bmatrix}\mathbf{H}_{11} & \mathbf{H}_{12}\\ \mathbf{H}_{21} & \mathbf{H}_{22}\end{bmatrix}\begin{Bmatrix}\mathbf{u}_{\mathrm{B}}^{\mathrm{B}}\\ \mathbf{u}_{\mathrm{B}}^{\mathrm{I}}\end{Bmatrix}=\begin{Bmatrix}\mathbf{f}_{\mathrm{B}}^{\mathrm{B}}\\ \mathbf{f}_{\mathrm{B}}^{\mathrm{I}}\end{Bmatrix} \tag{4.48}$$

Where: $\{\mathbf{f}_{\mathrm{B}}^{\mathrm{B}}\},\{\mathbf{f}_{\mathrm{B}}^{\mathrm{I}}\}$ are non-interface and interface known right-side vectors, respectively for the BEM sub-domain,

And reduced second to the form: $[\mathbf{H}_{\mathbf{B}}^{*}]\left\{\mathbf{u}_{\mathrm{B}}^{\mathrm{I}}\right\}=\{\mathbf{f}_{\mathrm{B}}^{*}\}$ (4.49)

Where:

$$[\mathbf{H}_{\mathbf{B}}^{*}] = [\mathbf{H}_{22}] - [\mathbf{H}_{21}][\mathbf{H}_{11}^{-1}][\mathbf{H}_{12}] \tag{4.50}$$

$$\{\mathbf{f}_{\mathbf{B}}^{*}\} = \{\mathbf{f}_{\mathrm{B}}^{\mathrm{I}}\} - [\mathbf{H}_{21}][\mathbf{H}_{11}^{-1}]\{\mathbf{f}_{\mathrm{B}}^{\mathrm{B}}\} \tag{4.51}$$

The method algorithm consists of two main stages:

I. Preconditioning:

1. In advance, perform an additional Cholesky decomposition for matrix $[\mathbf{K}_{\mathbf{F}}^{*}]$, exploiting the symmetry property of $[\mathbf{K}_{\mathbf{11}}]$, and LU decomposition for matrix $[\mathbf{H}_{\mathbf{B}}^{*}]$,

2. Set initial guess for $\{\bar{\mathbf{f}}_{\mathrm{F},0}^{\mathrm{I}}\}$,

3. Find $\{\bar{\mathbf{t}}_{\mathrm{B},0}^{\mathrm{I}}\}$ using equation (4.27) and equilibrium condition,

4. Obtain $\{\mathbf{u}_{\mathrm{F},0}^{\mathrm{I}}\}$ in the FE sub-domain by solving equation (4.45),

5. Obtain $\{\mathbf{u}_{\mathrm{B},0}^{\mathrm{I}}\}$ in the BE sub-domain by solving equation (4.49),

6. Obtain the CG initial parameters on interface $\mathbf{r}^{\mathrm{I}}$ using the following equations:

$$g_{,0}^{i} = u_{\mathrm{B},0}^{i} - u_{\mathrm{F},0}^{e} \tag{4.51}$$

$$w_{,0}^{i} = g_{,0}^{i} \tag{4.52}$$

$$\mu_{,0}^{i} = \bar{f}_{,0}^{i} = -\bar{f}_{,0}^{e} \tag{4.53}$$

Where: i and e are boundary and finite element numbers, respectively,

$u_{\mathrm{B},0}^{i} = u_{\mathrm{F},0}^{e}\left(\equiv \dfrac{u_{\mathrm{F},0}^{k} + u_{\mathrm{F},0}^{l}}{2}\right)$, k and l are interface finite element node numbers,

The potential is constant on the BE and linear on the FE,

The flux is constant on the BE and on the FE.

II. Iteration:

For n = 1, 2, ... continue until convergence,

1. Apply the following:

$$\{\bar{\mathbf{u}}_B^B\} = \{\bar{\mathbf{u}}_F^F\} = 0 \quad \text{on } \mathbf{r}^B \tag{4.54}$$

$$\{\bar{\mathbf{t}}_B^B\} = \{\bar{\mathbf{t}}_F^F\} = 0 \quad \text{on } \mathbf{r}^F \tag{4.55}$$

$$\bar{f}_B^i = -\bar{f}_F^e = w_{,n}^i \quad \text{on } \mathbf{r}^I \tag{4.56}$$

2. Obtain $\{\mathbf{u}_{F,n}^I\}$ in the FE sub-domain by solving equation (4.45) and equation (4.55),

3. Obtain $\{\mathbf{u}_{B,n}^I\}$ in the BE sub-domain by solving equation (4.49) and equation (4.54),

4. Compute the residual vector $\{\boldsymbol{\delta}\mathbf{u}_{B,n}^I\}$ of the potential and the CG parameters:

$$\delta u_{,n}^i = u_{B,n}^i - u_{F,n}^e \tag{4.57}$$

$$\mu_{,n+1}^i = \mu_{,n}^i - \rho w_{,n}^i \tag{4.58}$$

$$w_{,n+1}^i = g_{,n+1}^i + \kappa w_{,n}^i \tag{4.59}$$

$$g_{,n+1}^i = g_{,n}^i + \rho \delta u_{,n}^i \tag{4.60}$$

Where:

$$\rho = \frac{\sum_{i=1}^{m} (g_{,n}^i)^2}{\sum_{i=1}^{m} w_{,n}^i \delta u_{,n}^i} \tag{4.61}$$

$$\kappa = \frac{\sum_{i=1}^{m} (g_{,n+1}^i)^2}{\sum_{i=1}^{m} (g_{,n}^i)^2} \tag{4.62}$$

i=1, ... m (m is the number of nodes on the interface),

5. Repeat the iteration steps until convergence. Convergence is achieved when:

$$\left|\mu^{i}_{,n+1} - \mu^{i}_{,n}\right| \le \varepsilon \tag{4.63}$$

Where: ε is a small predefined tolerance.

The disadvantage of the CG methods is that it only solves positive definite symmetric linear set of equations.

Dong [88] employed Ma and Le's [89] symmetrisation-iteration method (SIM) in developing an iterative BE-FE coupling method. In this method, the condensation technique is used to reduce the BE sub-domain's stiffness matrix. SIM solves the equations in the combined asymmetric matrix of the coupled BE-FE sub-domains and considers the influence of the asymmetry of the boundary element sub-domain's stiffness matrix. The convergence of Dong's method depends on the choice of an enhanced symmetric matrix. The combination of equations (4.17) and (4.25) produces the following equation:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} + \mathbf{K}^{I}_{B} \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{F}_{F} \\ \mathbf{u}^{I}_{F} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}^{F}_{F} \\ -\mathbf{f}^{I}_{B} \end{Bmatrix} \tag{4.64}$$

Where: $\mathbf{K}^{I}_{B} = \mathbf{N}\mathbf{K}_{BE}$ and $\mathbf{f}^{I}_{B} = \{\mathbf{F}\}_{co} + \mathbf{N}\{\mathbf{t}\}_{co}$

To save the symmetry and sparsity of the global stiffness matrix in equation (4.64), Dong put matrix $\mathbf{K}^{I}_{B}$ in the right-side of this equation and introduced the enhanced symmetric matrix $\mathbf{K}_{1}$ into the same equation as the following:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} + \mathbf{K}_{1} \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{F}_{F} \\ \mathbf{u}^{I}_{F} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}^{F}_{F} \\ -\mathbf{f}^{I}_{B} \end{Bmatrix} + \begin{Bmatrix} 0 \\ \mathbf{K}^{I}_{B} + \mathbf{K}_{1} \end{Bmatrix} \begin{Bmatrix} \mathbf{u}^{I}_{F} \end{Bmatrix} \tag{4.65}$$

Matrix $\mathbf{K}_{1}$ should be selected from the following options to achieve convergence:

i. $\mathbf{K}_{\mathrm{l}} = \mathbf{I}$ (the identity matrix),

ii. $\mathbf{K}_{\mathrm{l}}$ refers to the diagonal terms of $\mathbf{K}_{\mathrm{B}}^{\mathrm{I}}$,

iii. $\mathbf{K}_{\mathrm{l}} = \frac{1}{2}\{[\mathbf{K}_{\mathrm{B}}^{\mathrm{I}}]^{\mathrm{T}} + [\mathbf{K}_{\mathrm{B}}^{\mathrm{I}}]\}$

The compacted form of equation (4.65), to be solved iteratively, is given by:

$$\overline{\mathbf{K}}\mathbf{u}_{\mathrm{n+1}} = \overline{\mathbf{F}} + \overline{\mathbf{K}}_{\mathrm{I}}\mathbf{u}_{\mathrm{n}} \tag{4.66}$$

Where: $\overline{\mathbf{K}} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} + \mathbf{K}_{\mathrm{l}} \end{bmatrix}$, $\overline{\mathbf{K}}_{\mathrm{I}} = \begin{bmatrix} 0 & 0 \\ 0 & -\mathbf{K}_{\mathrm{B}}^{\mathrm{I}} + \mathbf{K}_{\mathrm{l}} \end{bmatrix}$ and $\overline{\mathbf{F}} = \begin{Bmatrix} \mathbf{f}_{\mathrm{F}}^{\mathrm{F}} \\ -\mathbf{f}_{\mathrm{B}}^{\mathrm{I}} \end{Bmatrix}$

The iteration scheme starts with setting an initial value $\mathbf{u}_0$, then solving equation (4.66) for a sequence of displacements. The iteration stops whenever the following iteration condition is satisfied:

$$\frac{\left\|\mathbf{u}_{\mathrm{n+1}} - \mathbf{u}_{\mathrm{n}}\right\|}{\left\|\mathbf{u}_{\mathrm{n+1}}\right\|} \leq \varepsilon \tag{4.67}$$

Feng and Owen [57] created an iterative FE/BE coupling scheme to find the solution for a plate resting on an elastic half space foundation problem. The BEM and the FEM are used to analyze the elastic homogeneous half space medium, and the non-linear inhomogeneous plate behaviours, respectively. The equations, which represent the FE and BE sub-domains, are rewritten using the authors' (Feng and Owen) symbols as the following:

$$[\mathbf{K}_{\mathrm{p}}]\{\mathbf{U}\} = \{\mathbf{F}\} - \{\mathbf{R}\} \quad \text{/ FE sub-domain /} \tag{4.68}$$

Where: $[\mathbf{K}_{\mathrm{p}}]$ is the plate stiffness matrix, $\{\mathbf{U}\} = \{\mathbf{W}_{\mathrm{p}}, \boldsymbol{\theta}_{\mathrm{x}}, \boldsymbol{\theta}_{\mathrm{y}}\}^{\mathrm{T}}$ is the nodal unknowns vector of the plate, $\{\mathbf{F}\}$ is the plate nodal external applied forces vector, $\{\mathbf{R}\}$ is the interactive nodal forces vector between the plate and the foundation.

$$\{\mathbf{W}_f\} = [\mathbf{G}]\{\mathbf{P}\} \quad \text{/ BE sub-domain /} \tag{4.69}$$

Where: [**G**] is the coefficient matrix using Boussinesq solution, $\{\mathbf{W}_f\}$ and $\{\mathbf{P}\}$ are the nodal vertical displacements and nodal traction vectors of the foundation surface. Only the plate-foundation interface is discretized into boundary elements using this method.

The sequential Dirichlet-Neumann FEM-BEM coupling method algorithm is:

For n = 0, 1, 2, ... continue/repeat until convergence,

1. Set initial guess$\{\mathbf{R}_0\}$,

2. Obtain $\{\mathbf{U}_{n+1}\}$ by solving the equation:

$$[\mathbf{K}_p]\{\mathbf{U}_{n+1}\} = \{\mathbf{F}\} - \{\mathbf{R}_n\} \tag{4.70}$$

3. Obtain the plate nodal vertical displacement $\{\mathbf{W}_{p,n+1}\}$ by:

$$\{\mathbf{W}_{p,n+1}\} = \{\mathbf{E}\}^T\{\mathbf{U}_{n+1}\} \tag{4.71}$$

Where: $\{\mathbf{E}\} = [\mathbf{I}_m, 0, 0]^T$ is a matrix to extract the vector $\{\mathbf{W}_{p,n+1}\}$ from $\{\mathbf{U}_{n+1}\}$.

4. Apply compatibility condition:

$$\{\mathbf{W}_{f,n+1}\} = \{\mathbf{W}_{p,n+1}\} \tag{4.72}$$

5. Obtain $\{\mathbf{P}_{n+1}\}$ by solving the equation:

$$[\mathbf{G}]\{\mathbf{P}_{n+1}\} = \{\mathbf{W}_{f,n+1}\} \tag{4.73}$$

6. Apply the equilibrium condition and find $\{\mathbf{R}_{n+1}\}$ using the following equation:

$$\{\mathbf{R}_{n+1}\} = \{\mathbf{E}\}[\mathbf{M}]\{\mathbf{P}_{n+1}\} \tag{4.74}$$

Where: [**M**] is the converting matrix, defined earlier.

7. Return to step (2) and repeat the steps that follow until convergence.

Convergence is achieved when:

$$\frac{\left\|\mathbf{U}_{\mathbf{n}+1}-\mathbf{U}_{\mathbf{n}}\right\|}{\left\|\mathbf{U}_{\mathbf{n}+1}\right\|}<\varepsilon \tag{4.75}$$

Where: ε is a small predefined tolerance.

Feng and Owen [57] modified equation (4.70) to speed up the convergence and decrease the iterations' number, to have the following form:

$$\{\mathbf{U}_{n+1}\}=[\mathbf{B}]\{\mathbf{U}_{n}\}+\{\mathbf{b}\} \tag{4.76}$$

Where:

$$[\mathbf{B}]=-([\mathbf{K}_{p}]+\alpha[\overline{\mathbf{K}}_{\mathbf{L}}])^{-1}\{\mathbf{E}\}([\mathbf{K}_{f}]-\alpha[\mathbf{K}_{\mathbf{L}}])\{\mathbf{E}\}^{T} \tag{4.77}$$

$$\{\mathbf{b}\}=([\mathbf{K}_{p}]+\alpha[\overline{\mathbf{K}}_{\mathbf{L}}])^{-1}\{\mathbf{F}\} \tag{4.78}$$

$$[\overline{\mathbf{K}}_{\mathbf{L}}]=\{\mathbf{E}\}[\mathbf{K}_{\mathbf{L}}]\{\mathbf{E}\}^{T} \tag{4.79}$$

$$[\mathbf{K}_{f}]=[\mathbf{M}][\mathbf{G}]^{-1} \tag{4.80}$$

α is a relaxation parameter, matrix $[\mathbf{K}_{L}]$ is an approximation to $[\mathbf{K}_{f}]$. α and $[\mathbf{K}_{L}]$ are selected by the user according to a specific criterion derived and recommended by Feng and Owen [57]. In spite of these recommendations, the convergence of this method still depends on the selected: parameter α, matrix $[\mathbf{K}_{L}]$ and the tolerance ε.

Lin et al. [51] developed an iterative FE/BE coupling method. This non overlapping domain decomposition method is based on what Funaro et al. [90] suggested for solving second-order elliptic problems iteratively using the spectral collocation approximation, and an interface relaxation approach with automatic selection of the relaxation parameter at each iteration. The sequential Dirichlet-Neumann coupling method algorithm is:

For n =0, 1, 2, ... do until convergence,

1. Set initial guess for $\{\mathbf{u}^{I}_{B,0}\}$,

2. Obtain $\{\mathbf{f}^{\mathbf{I}}_{B,n}\}$ in the BE sub-domain by solving equation (4.26) and by using equation: $\{\mathbf{f}^{I}_{B,n}\} = [\mathbf{M}]\{\mathbf{t}^{I}_{B,n}\}$,

3. Apply: $\{\mathbf{f}^{\mathbf{I}}_{F,n}\} = -\{\mathbf{f}^{\mathbf{I}}_{B,n}\}$ (4.81)

4. Obtain $\{\mathbf{u}^{\mathbf{I}}_{F,n}\}$ in the FE sub-domain by solving equation (4.25),

5. Check the solution convergence if:

$$\left\|\mathbf{u}^{I}_{F,n} - \mathbf{u}^{I}_{B,n}\right\| \le \varepsilon \quad (4.82)$$

   Stop if the convergence condition is satisfied; otherwise, proceed to the next step.

6. Apply:

$$\{\mathbf{u}^{I}_{B,n+1}\} = \omega\{\mathbf{u}^{I}_{F,n}\} + (1-\omega)\{\mathbf{u}^{I}_{B,n}\}, \text{ for } n \ge 1 \quad (4.83)$$

   Where: ω is a relaxation parameter.

7. Return to step (2) and repeat the steps that follow until convergence.

The relaxation parameter ω is selected by the user in the range $0 < \omega \le 1$. Because there are infinite number of values for the relaxation parameter, Lin et al. [51] propose a dynamic method to obtain the optimal value for ω, which reduces the iteration numbers to the minimum. The optimal value of ω, determined by minimizing the square error function of two consecutive iteration values of the interfacial displacement, is given by:

$$\omega = \frac{\left(\mathbf{e}^{B}_{n}, \mathbf{e}^{B}_{n} - \mathbf{e}^{F}_{n}\right)}{\left\|\mathbf{e}^{B}_{n} - \mathbf{e}^{F}_{n}\right\|^{2}}, \text{ for } n \ge 2 \text{ and } 0 < \omega \le 1 \quad (4.84)$$

Where: $\mathbf{e}_n^B = \mathbf{u}_{B,n}^I - \mathbf{u}_{B,n-1}^I$ and $\mathbf{e}_n^F = \mathbf{u}_{F,n}^I - \mathbf{u}_{F,n-1}^I$ (4.85)

Although Lin et al. [51] emphasised that the initial guess for the interfacial displacement may take any value, the authors suggest starting with the zero value. This assumption, which seems physically reasonable, corresponds to the case of fixed interface.

Elleithy et al. [41]utilized the iterative FE/BE coupling method developed by Lin et al. [51] and used the same sequential Dirichlet-Neumann coupling algorithm. The algorithm starts with imposing an initial nodal displacement vector on the interface for the BEM sub-domain. The solved nodal traction vector is applied, converted into nodal forces vector, according to the equilibrium condition on the interface as a BC for the FEM sub-domain in the next step. Then if the solved and the assumed interfacial displacement vectors for both domains do not satisfy the convergence condition, the iteration continues until convergence is achieved. Elleithy et al. [41] established two conditions (equations 4.86 & 4.87); this iterative method should fulfill to achieve the convergence. Selecting the relaxation parameter ω seen in equation (4.83) beyond the limits of these conditions will not lead the iterative method of Lin et al. [51] to the targeted convergence.

$$\omega < \min_{1\le i\le N}\left\{\frac{2(1-x_i)}{(1-x_i)^2+y_i^2}\right\} \tag{4.86}$$

$$\text{and} \quad x_i < 1, i = 1,2,.....N \tag{4.87}$$

Where:

$\lambda_1 = x_1+iy_1$, $\lambda_2 = x_2+iy_2$,..., $\lambda_N = x_N+iy_N$ are the eigenvalues of matrix C, matrix C is given by:

$$\mathbf{C} = -\mathbf{F}_{22}\mathbf{M}\mathbf{A}_{22} \tag{4.88}$$

and matrices $\mathbf{F}_{22}$ and $\mathbf{A}_{22}$ can be seen defined in the rearranged form of equations (4.25) and (4.26) as:

$$\begin{Bmatrix} \mathbf{u}_F^F \\ \mathbf{u}_F^I \end{Bmatrix} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{C}_F \\ \mathbf{f}_F^I \end{Bmatrix} \tag{4.89}$$

$$\begin{Bmatrix} \mathbf{X}_B^B \\ \mathbf{t}_B^I \end{Bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{C}_B \\ \mathbf{u}_B^I \end{Bmatrix} \tag{4.90}$$

Where: $\mathbf{X}_B^B$ are the non-interface boundary unknowns in BE sub-domain, $\mathbf{C}_B$ and $\mathbf{C}_F$ are vectors of constant values.

"One can conclude that convergence is dependent on the eigenvalues of matrix **C**, which in turn are dependent on **K**, **H**, **G** and **M** matrices. This indicates that convergence is dependent on the mesh density of the problem sub-domains, specified boundary conditions, and the geometrical and material properties of the sub-domains." [41]

Non overlapping domain decomposition iterative coupling is employed to couple the non-linear plastic behaviour of the FEM sub-domain with the linear elastic one of the BEM sub-domain by Perera et al. [81] and Elleithy and Tanaka [70] and others. Elleithy and Tanaka explicitly outlined the method's sequential Dirichlet-Neumann algorithm as the following:

The incremental form of equation (4.25) is rewritten first to analyze the non-linear elasto-plastic behaviour of the FEM sub-domain as:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{u}_F^F \\ \Delta\mathbf{u}_F^I \end{Bmatrix} = \begin{Bmatrix} \Delta\mathbf{f}_F^F \\ \Delta\mathbf{f}_F^I \end{Bmatrix} \tag{4.91}$$

Set initial guess for $\{\mathbf{u}_{B,0}^I\}$,

For n =0, 1, 2,...continue/repeat until convergence,

1. Obtain $\{\mathbf{t}_{B,n}^I\}$ in the BE sub-domain by solving equation (4.26),

2. Obtain $\mathbf{\{f_{F,n}^I\}}$ using equation $\mathbf{\{f_{F,n}^I\}} + \mathbf{[M]\{t_{B,n}^I\}} = 0$,

For i = 1, 2, ... Specified number of increments,

2.1. Obtain $\{\Delta \mathbf{u}^{I}_{F,i}\}_n$ in the FE sub-domain by solving equation (4.91),

2.2. Apply $\{\Delta \mathbf{u}^{I}_{F,i+1}\}_n = \{\mathbf{u}^{I}_{F,i}\}_n + \{\Delta \mathbf{u}^{I}_{F,i}\}_n$ (4.92)

2.3. Obtain $\{\mathbf{u}^{I}_{F,n}\}$

3. Apply:

$$\{\mathbf{u}^{I}_{B,n+1}\} = \{\mathbf{u}^{I}_{B,n}\} + \gamma(\{\mathbf{u}^{I}_{F,n}\} - \{\mathbf{u}^{I}_{B,n}\})$$

Where: γ is a relaxation parameter.

4. Repeat the steps from 1 to 4 until convergence.

4.3 Coupling BEM with FDM

Coupling the finite difference method (FDM) with the boundary element method (BEM) is developed in the mid1960s to solve different problems in different fields [43]. These problems can be classified mainly into two groups:

The first group is the time-independent group of problems, such as elasto-plasticity [93-96], elliptic diffusion (Laplace) [45], [97], [98], [117], [118], and elasto-hydrodynamic [115], [116] problems. The second group is the time-dependent set of problems, such as parabolic diffusion [5], [99], [103], [119], [120], viscous flow (governed by Navier-Stokes equations) [104], [110], dynamics [111], [114], and similar time-dependent problems. Each group may be divided into two classes, depending on the method of coupling BEM with FDM: the first class of problems is solved by coupling BEM with FDM at the level of discretized equations [5], [93], [97], [99-102], [104-108], [111] and the second class is solved by coupling BEM with FDM using the domain decomposition method with an iterative scheme [98], [116-120]. Although coupling the FD/BE method has been developed, in its early beginnings, to solve different types of

problems such as elasto-plasticity problems, the method excelled significantly in the last two decades in solving the diffusion, dynamics, acoustic and many other time-dependent problems.

4.3.1 Coupling BEM with FDM at the Level of Discretized Equations

The discretized equations for the BEM and the FDM sub-domains, in this class, are combined to build an entire unified system of equations for the whole domain. The following two research works are study cases chosen from the existing time-independent FDM/BEM coupling at the level of discretized equations [93-97].

Banerjee and Davies [94] worked on solving soil-interaction elasto-plastic problems using a combination of the BEM for the semi-infinite soil sub-domain and the FDM for a group of piles with a cap. Boundary Integral Equation for the soil sub-domain is simplified to have the form:

$$\mathbf{u} = \int_S \mathbf{G}\tilde{\boldsymbol{\varphi}}\,d\mathbf{S} \tag{4.93}$$

Where: $\tilde{\boldsymbol{\varphi}}$ is a fictitious traction; nearly identical to real surface traction at S (the pile-soil interface surface) for pile slenderness ratios greater than 5, **u** is the interface nodal displacement vector, and **G** is Mindlin's fundamental solution matrix.

The boundary (the pile-soil interface) is discretized into number of boundary elements. $\tilde{\boldsymbol{\varphi}}$ varies linearly over these elements. Hence, Equation (4.93) rewritten in matrix and vector format becomes:

$$\mathbf{U} = \mathbf{K}\tilde{\boldsymbol{\Phi}} \tag{4.94}$$

Where: **K** is the influence coefficient, **U** and $\tilde{\boldsymbol{\Phi}}$ are the global interface nodal displacement and traction vectors, respectively. The group of piles domain is represented, using FDM and taking into consideration the compressibility and the flexural rigidity of the piles, by the following equation:

$$\mathbf{U} = \mathbf{D}\tilde{\mathbf{\Phi}} + \mathbf{b} \tag{4.95}$$

Where: **D** is a matrix of finite difference coefficients, and **b** is the boundary condition vector.

The BEM and FDM are coupled at the level of discretized equations. The combined system of equation (4.94) and equation (4.95), after satisfying the compatibility and equilibrium conditions, is given by:

$$\mathbf{(K - D)}\tilde{\mathbf{\Phi}} = \mathbf{b} \tag{4.96}$$

Rangogni and Reali [97] developed a coupling BEM/FDM technique to solve two-dimension-Laplace problems defined on a domain Ω with a boundary Γ by:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad (x, y) \in \Omega \tag{4.97}$$

$$u(x, y) = f(x, y) \qquad (x, y) \in \Gamma_1 \tag{4.98}$$

$$\frac{\partial u(x, y)}{\partial n} = g(x, y) \qquad (x, y) \in \Gamma_2 \text{ with } \Gamma_1 \bigcup \Gamma_2 = \Gamma \tag{4.99}$$

Where: $\frac{\partial u(x, y)}{\partial n}$ is the normal derivative of u on Γ, f(x, y) and g(x, y) are given functions.

Laplace differential equation is transformed into a set of linear system of equations for u_i values of u at all points P_i(i=1,2....N) in Ω_1, the FDM sub-domain. The derivatives seen in the PDE are approximated using a general finite difference forms (FDF):

$$\left[\frac{\partial u}{\partial s}\right]_{p_i} = \sum_k \alpha_{i,k} u_{i,k} + \alpha_i u_i \tag{4.100}$$

Where: $\left[\frac{\partial u}{\partial s}\right]_{p_i}$ is u derivative in s direction at P_i, $u_{i,k}$ are the values of u at suitably chosen grid points $P_{i,k}$ about grid point P_i, $\alpha_{i,k}$ and α_k are coefficients depending on the coordinates of grid

points $P_{i,k}$ and P_i. Number of terms (k) in the sum depends on the chosen order of approximation.

Boundary integral equation over boundary Γ of Ω_2, the BEM sub-domain is:

$$u(P) = \int_{\Gamma} \frac{\partial}{\partial n} G(P,Q).u(Q)d\Gamma - \int_{\Gamma} G(P,Q)\frac{\partial u}{\partial n}(Q)d\Gamma \qquad (4.101)$$

G(P,Q) is the Green function given by:

$$G(P,Q) = \frac{1}{\eta(P)} \ln|P-Q| \qquad (4.102)$$

Where: $\eta(Q)$ is the geometrical coefficient.

The discretized form of equation (4.101), for boundary Γ divided into the number of boundary elements and (k) boundary nodes have the general expression:

$$u_i = \sum_k \lambda_k u_k \sum_k \delta_k u'_k \qquad (4.103)$$

Where: λ_k and δ_k are the influence coefficients, u'_k is the value of $\left[\frac{\partial u}{\partial n}\right]_{p_k}$ at node P_k.

The system of linear equations for FDM sub-domain Ω_1 obtained from equations (4.97-100) and the system of equations(4.103) for BEM sub-domain Ω_2 are combined and solved for u on interface Γ_3 and boundary Γ_1 and for $\left[\frac{\partial u}{\partial n}\right]$ on boundary Γ_2.

The following three research works are solving time-dependent problems using FDM/BEM coupling at the level of discretized equations [5], [99], [104-108], [111]. These coupling approaches can be reclassified into two groups: In the first group there is no spatial discretization of the domain, but the time derivatives that appear in these coupling approaches' PDE(s) are approximated using a finite difference formulation and the equations' solution advances in time using the introduced FD formulation [5], [99-102], [104-108]. In the second

group spatial discretizations in the interacting sub-domains are introduced [111] and an entire unified system of equations for the whole domain is built and solved.

Brebbia et al. [5] presented a coupled BE/FD method to solve the diffusion problems. The time derivative that appears in the diffusion equation (4.104) is approximated in a finite difference form. The diffusion equation, the associated boundary conditions (4.105), and the initial condition (4.106) are given as:

$$\nabla^2 u(Q,t) - \frac{1}{k}\frac{\partial u(Q,t)}{\partial t} = 0 \qquad Q \in \Omega \tag{4.104}$$

$$\left.\begin{aligned} u(Q,t) &= \bar{u}(Q,t) && Q \in \Gamma_1 \\ q(Q,t) &= \frac{\partial u(Q,t)}{\partial n(Q)} = \bar{q}(Q,t) && Q \in \Gamma_2 \end{aligned}\right\} \tag{4.105}$$

$$u(Q,t) = u_0(Q_0,t_0) \qquad Q \in \Omega \tag{4.106}$$

Where: k is a constant parameter.

Equation (4.104), using the Laplace transform of function u(Q, t) into U(Q, λ), becomes:

$$\nabla^2 U(Q,\lambda) - \frac{\lambda}{k}U(Q,\lambda) + \frac{1}{k}u_0(Q_0,t_0) = 0 \tag{4.107}$$

Where: λ is the transform parameter.

The boundary integral equation (BIM) of the problem is given by:

$$c(P)U(P,\lambda) + k\int_\Gamma U(Q,\lambda)Q^*(P,Q,\lambda)d\Gamma(Q) = k\int_\Gamma Q(Q,\lambda)U^*(P,Q,\lambda)d\Gamma(Q) + \int_\Omega u_0(Q,t_0)U^*(P,Q,\lambda)d\Omega(Q)$$

Where: $Q^*(P,Q,\lambda) = \dfrac{\partial U^*(P,Q,\lambda)}{\partial n(Q)}$ and $Q(P,Q,\lambda) = \dfrac{\partial U(P,Q,\lambda)}{\partial n(Q)}$ (4.108)

The fundamental solution U^* for three-dimensional and two-dimensional problems, respectively are:

$$U^* = \frac{(k\lambda)^{1/4}}{r^{1/2}(2\pi k)^{3/2}} K_{1/2}\left[\left(\frac{\lambda}{k}\right)^{1/2} r\right] \tag{4.109}$$

$$U^* = \frac{1}{2\pi k} K_0\left[\left(\frac{\lambda}{k}\right)^{1/2} r\right] \tag{4.110}$$

Where: K_v is the modified Bessel function of the second kind of order v.

The time derivative of the solution u(Q,t), approximated by a finite difference form, is given for a Δt time step as:

$$\frac{\partial u(Q,t)}{\partial t} = \frac{u(Q,t+\Delta t) - u(Q,t)}{\Delta t} \tag{4.111}$$

Equation (4.104) becomes:

$$\nabla^2 u(Q,t+\Delta t) - \frac{1}{k\Delta t} u(Q,t+\Delta t) + \frac{1}{k\Delta t} u(Q,t) = 0 \tag{4.112}$$

The above equation is very similar to equation (4.107); therefore the BIE of the problem defined in equation (4.112) is similar to the boundary integral in equation (4.108):

$$\begin{aligned} &c(P)U(P,t+\Delta t) + k\int_\Gamma U(Q,t+\Delta t) q^*(P,Q,\Delta t) d\Gamma(Q) \\ &= k\int_\Gamma Q(Q,t+\Delta t) u^*(P,Q,\Delta t) d\Gamma(Q) + \frac{1}{\Delta t}\int_\Omega u(Q,t) u^*(P,Q,\Delta t) d\Omega(Q) \end{aligned} \tag{4.113}$$

The fundamental solutions u* and q* are the same as those defined in equations (4.109) and (4.110), but after replacing λ with $1/\Delta t$.

Wu and Thomson [104] numerically solved the incompressible viscous flow, governed by Navier-Stokes equations, problems using a developed coupled BE/FD method. The vorticity transport equation, in this method, is solved using an FDM and the kinematic vorticity equation is solved using a BEM. The Navier-Stokes equation is written in vector notations as the following:

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}.\nabla)\mathbf{v} = -\frac{1}{\rho}\nabla \mathbf{p} + \nu \nabla^2 \mathbf{v} \tag{4.114}$$

Where: **v** and **p** are the flow velocity and the pressure, ρ and υ are the mass density and the viscosity of the fluid, respectively.

The curl of equation (4.114) gives the vorticity transport equation:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \nabla \times (\mathbf{v} \times \boldsymbol{\omega}) = \nu \nabla^2 \boldsymbol{\omega} \tag{4.115}$$

The vorticity **ω** is given by:

$$\nabla \times \mathbf{v} = \boldsymbol{\omega} \tag{4.116}$$

The kinematic of the flow is represented by the following Boundary integral equation [5], [104]

$$\mathbf{v}(P) = \frac{1}{2\alpha\pi}\left[\begin{array}{l} \int_V \frac{\boldsymbol{\omega}(Q)\times \mathbf{r}(P,Q)}{r^d(P,Q)} dV(Q) + \int_S \frac{[\mathbf{v}(Q)\times \mathbf{n}(Q)]\times \mathbf{r}(P,Q)}{r^d(P,Q)} dS(Q) \\ + \int_S \frac{[\mathbf{v}(Q).\mathbf{n}(Q)]\mathbf{r}(P,Q)}{r^d(P,Q)} dS(Q) \end{array}\right] + \mathbf{B} \tag{4.117}$$

Where: Volume V is bounded by the surface S, S consists of either a single closed surface or a closed outer surface and an inner closed surface (s), α=2 and d=3 for 3D problems and α=1 and d=2 for 2D problems. Vector **B** vanishes for flows interior of S and equals $\mathbf{v}_\infty$ for flows exterior of S, $\mathbf{v}_\infty$ is the constant stream velocity at infinity.

Because the velocity at the fluid-solid interface equals zero (no slip condition) for problems of external flow past finite bodies, the integrals in equation (4.117) become:

$$\mathbf{v}(P) = \frac{1}{2\alpha\pi}\left[\int_V \frac{\boldsymbol{\omega}(Q)\times \mathbf{r}(P,Q)}{r^d(P,Q)} dV(Q)\right] + \mathbf{v}_\infty \tag{4.118}$$

The flow domain volume V is divided into cells. The numerical computations start with known values of **ω** (**r**, t) and **v** (**r**, t). In the first step values of **ω** (**r**, t + Δt) are computed for each cell using a finite difference representation, described by Thompson [109], of the vorticity transport

equation (4.115). In the second step the values of **v** (**r**, t + Δt) for each cell is computed using a discretized form of the kinematic vorticity equation (4.118) and the computed values of **ω** (**r**, t + Δt). And in the third step using the values of **v** (**r**, t + Δt) and **ω** (**r**, t + Δt) in the boundary cells, boundary conditions, and a finite difference representation of equation (4.116), values of **ω** (**r**, t + Δt) for the boundary cells are computed.

Maamoon and Banerjee [111] developed a coupled BEM/FDM technique to solve an elasto-dynamic problem in the geotechnical field. The numerical analysis of a single pile response under time-dependent forces and wave excitation is performed using the developed coupled method at the level of discretized equations. Displacement in the soil elastic-half-space sub-domain is obtained using the BEM with step-by-step time integration scheme. The differential equation for dynamics in homogenous, isotropic, elastic solid is given by:

$$(\lambda+\mu)u_{p,pq}+\mu u_{q,pp}-\rho\ddot{u}_q=0 \tag{4.119}$$

Where: λ and μ are Lame's constants, ρ is the mass density of the solid, $\ddot{u}_q=\frac{\partial^2 u_q}{\partial t^2}$ is the acceleration.

The transient response in BEM sub-domain at time t_j is obtained using the following BIE in terms of Green function G_{pq} for half-space:

$$u_p(P,t_j)-\int_{\tau=t_j-1}^{t_j}\int_S G_{pq}\Phi_q\,ds\,d\tau=\int_{\tau=0}^{t_{j-1}}\int_S G_{pq}\Phi_q\,ds\,d\tau \tag{4.120}$$

Where: Φ_q is the traction at the pile-soil interface S.

The integral on the right hand side is the contribution due to the past dynamic history. If the actual distribution of the boundary tractions is assumed to be constant over the boundary elements, equation (4.120) written in vector and matrix form becomes:

$$\mathbf{U}_j^S = \int_{\tau=t_j-1}^{t_j} \mathbf{G}\boldsymbol{\Phi} d\tau + \int_{\tau=0}^{t_{j-1}} \mathbf{G}\boldsymbol{\Phi} d\tau \tag{4.121}$$

Where: $\mathbf{U}_j^S$ is the soil displacement at time step j, s stands for soil.

The integration over time in equation (4.121) is carried out analytically by averaging the displacements and tractions over at the two time nodes of an interval. Equation (4.121) becomes in discretized form as:

$$\mathbf{U}_j^S = \mathbf{G}_1\boldsymbol{\Phi}_j^S + \sum_{i=1}^{j-1} \mathbf{G}_{j-i+1}\boldsymbol{\Phi}_i^S = \mathbf{G}_1\boldsymbol{\Phi}_j^S + \mathbf{R}_j^S \tag{4.122}$$

Where: $\mathbf{G}_1$ is the coefficient obtained at the first time step, $\mathbf{R}_j^S$ is the effects of past time histories on the current time.

The pile sub-domain displacement, represented by compressible and flexible one-dimensional elements, is approximated by discretizing the differential equations of pile motions using the FDM. The pile is divided into (n) segments of length (δ) and the loading is applied over number of time increments. The axial pile response at time t_j is governed by the equation:

$$m\frac{\partial^2 u_j^z}{\partial t^2} - E_p A_p \frac{\partial^2 u_j^z}{\partial z^2} = -\pi d\Phi_j^z \tag{4.123}$$

Where: m is mass per unit length of the pile; u_j^z is the axial pile displacement at time t_j; E_p is Young's modulus; A_p is the pile cross section; d is the pile diameter and Φ_j^z is the axial traction on the pile at t_j; and Δt is a single time increment.

Time and space derivatives in equation (4.123) are approximated using backward and central difference formulations:

$$\varepsilon(\mathbf{D}_0^z)\mathbf{U}_j^z = \boldsymbol{\Phi}_j^z - 5\gamma(\mathbf{U}_{j-1}^z) + 4\gamma(\mathbf{U}_{j-2}^z) - \gamma(\mathbf{U}_{j-3}^z) - \mathbf{L}_j^z = \boldsymbol{\Phi}_j^z + \mathbf{B}_j^z \tag{4.124}$$

Where: $\mathbf{D}_0^z$ is the axial pile coefficient matrix, $\mathbf{B}_j^z$ a known vector, represents the effect of past dynamic history, $\mathbf{L}_j^z = \{(P_j / \pi d\delta), 0, ..., 0\}^T$, P_j is the load value on the pile at jth time step, $\varepsilon = \dfrac{A_p E_p}{\pi d\delta^2}$, and $\gamma = \dfrac{m}{\pi d(\Delta t)^2}$.

Equation (4.124) can be rewritten as:

$$\mathbf{U}_j^z = \mathbf{D}_A^z \mathbf{\Phi}_j^z + \mathbf{C}_j^z \tag{4.125}$$

Where: $\mathbf{D}_A^z = \dfrac{1}{\varepsilon}(\mathbf{D}_0^z)^{-1}$ and $\mathbf{C}_j^z = \dfrac{1}{\varepsilon}(\mathbf{D}_0^z)^{-1}\mathbf{B}_j^z$

The transverse pile response at time t_j is governed by the flexibility equation:

$$m\frac{\partial^2 u_j^x}{\partial t^2} + E_p I_p \frac{\partial^4 u_j^x}{\partial z^4} = -d\Phi_j^x \tag{4.126}$$

The discretized form of the above differential equation using an FD formulation is given by:

$$\kappa(\mathbf{D}_0^x)\mathbf{U}_j^x = -\mathbf{\Phi}_j^x + 5\eta(\mathbf{U}_{j-1}^x) - 4\eta(\mathbf{U}_{j-2}^x) + \eta(\mathbf{U}_{j-3}^x) - \mathbf{L}_j^x = -\mathbf{\Phi}_j^x + \mathbf{B}_j^x \tag{4.127}$$

Where: Φ_j^x is the transverse traction on the pile at t_j, I_p second moment of area of the pile cross section, $\mathbf{D}_0^x$ is the transverse pile coefficient matrix, $\mathbf{L}_j^x = \{(2H_j / d\delta), 0, ..., 0\}^T$, H_j is the horizontal load value on the pile at jth time step, $\kappa = \dfrac{E_p I_p}{d\delta^4}$, and $\eta = \dfrac{m}{d(\Delta t)^2}$. Equation (4.127) becomes:

$$\mathbf{U}_j^x = \mathbf{D}_T^x \mathbf{\Phi}_j^x + \mathbf{C}_j^x \tag{4.128}$$

Where: $\mathbf{D}_T^x = \dfrac{1}{\kappa}(\mathbf{D}_0^x)^{-1}$ and $\mathbf{C}_j^x = \dfrac{1}{\kappa}(\mathbf{D}_0^x)^{-1}\mathbf{B}_j^x$

The total pile response is obtained by combining equations (4.125) and (4.128):

$$\mathbf{U}_j^p = \mathbf{D}\mathbf{\Phi}_j^p + \mathbf{C}_j^p \tag{4.129}$$

The tractions acting at the pile at jth time step is obtained by combining both systems of equation (4.129) for the FD pile sub-domain and equation (4.122) for the BE soil sub-domain, and by satisfying equilibrium and compatibility conditions at the pile-soil interface as:

$$\mathbf{\Phi}_j^p = (\mathbf{G}_1 + \mathbf{D})^{-1}\mathbf{F}_j^p \qquad (4.130)$$

Where: $\mathbf{F}_j^p = \mathbf{R}_j^s - \mathbf{C}_j^p$

4.3.2 Iterative Coupling of BEM with FDM

Coupling BEM with FEM using iterative domain decomposition techniques were used and studied relatively intensively in the last two decades as discussed before in section (4.2.2). Conversely, coupling BEM with FDM, using mainly a non-overlapping interface relaxation domain decomposition method with iterative scheme, is developed recently [98], [116-120]. The following three research works [116-118] solved a time-independent diffusion problem, by dividing the domain into a FDM and BEM sub-domains. Whereas, the fourth research work [119], presented in this section, solved a time-dependent diffusion problem by dividing the domain into multi-BEM-sub-domains, and the time derivatives in the PDE (s) for each sub-domain are discretized using a FD form. Although these coupling approaches in these research works are case specific methods but it evades the disadvantages of coupling FDM with BEM at the level of discretized equations.

Barrett et al. [116] employed the coupled BE-FD method in the elasto-hydrodynamic non-linear numerical analysis of journal bearings, used in the field of automotive industry. Because, the elastic deformation of the contact region of the bearings (solved by BEM) affects the hydrodynamic pressure and the oil film thickness, the bearings' behaviour is highly non-linear. The BEM, used to solve the structural differential equations, is coupled with the FDM, used to solve Reynolds's equation for laminar fluid flow. The boundary element sub-domain is

represented by equation (4.26), which relates the contact surface nodal tractions with the nodal deflections. On the other hand, the finite difference fluid sub-domain is governed by Reynolds's equation:

$$\frac{\partial}{\partial x}\left(h^3\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial x}\left(h^3\frac{\partial p}{\partial y}\right)=6\eta U\frac{\partial h}{\partial x}+12\eta V \qquad (4.131)$$

Where: η is fluid viscosity, h is the film thickness, p is the pressure, U and V are the fluid velocity components.

The space derivatives in Reynolds's equation are obtained by the following finite difference formulations:

$$\left(\frac{\partial h}{\partial x}\right)_{i,j}=\frac{h_{i+1,j}-h_{i-1,j}}{2\delta x} \quad \text{and} \quad \left(\frac{\partial^2 p}{\partial x^2}\right)_{i,j}=\frac{p_{i+1,j}-2p_{i,j}+p_{i-1,j}}{(\delta x)^2} \qquad (4.132)$$

In order to couple both systems of equations (4.26) and (4.131), the boundary element matrices in equation (4.26) are transformed into a finite-element-like stiffness matrix, condensed to the active nodes, and rotated into local transverse and normal directions. "The non-linear elasto-hydrodynamic calculation is carried out iteratively by recalculating the film thickness due to the deformation and using this to recalculate the film pressure. A 2D Newton-Raphson technique was used to find the next trial solution at each step." [116]

Peng et al. [117] utilized, the iterative coupling of the natural boundary element method with the finite difference method using the non-overlapping domain decomposition Dirichlet-Neumann alternating algorithm, to study the coupled Darcy-Stokes equations under a pressure difference. The domain Ω, truncated from the unbounded porous medium, is divided into sub-domain Ω_1 of porous medium with a radius R_1 and sub-domain Ω_2, a circular cavity with a radius R, as shown in Figure (4.3). The cavity's boundary Γ is the interface between both sub-domains. The free fluid flow in sub-domain Ω_2 is modelled by the incompressible Stokes

equations and approximated using the natural boundary element method. On the other hand, the flow in sub-domain Ω_1 of porous medium is governed by a discretized form of Darcy equations, using a finite difference formulation.

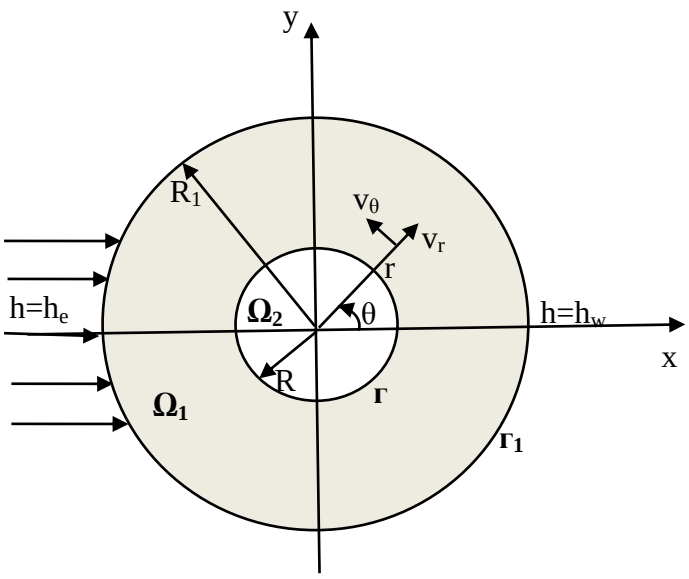


Figure 4.3 Domain sub-divisions and boundary conditions

Where: r and θ are polar coordinates; u_r, u_θ and p are radial velocity, tangential velocity and pressure of the free flow in the cavity; v_r, v_θ and h are radial velocity, tangential velocity and pressure of the flow in the porous medium; h_e and h_w are the entrance and exit pressure, respectively. The velocities and pressure are continuous across the interface Γ.

Darcy's law for small porosity, which governs the flow in Ω_1 sub-domain, is given by:

$$\frac{K}{\mu}\nabla^2 h = 0 \qquad (4.133)$$

Where: K and μ are the permeability coefficient and dynamic viscosity.

The finite difference approximation of equation (4.133) in terms of polar coordinates is:

$$\left(\frac{4(R+j\Delta r)^2\Delta\theta^2}{\Delta r^2}+4\right)h_{i,j}-\left(\frac{2(R+j\Delta r)^2\Delta\theta^2}{\Delta r^2}+\frac{(R+j\Delta r)\Delta\theta^2}{\Delta r}\right)h_{i,j+1}-2h_{i+1,j}$$
$$-\left(\frac{2(R+j\Delta r)^2\Delta\theta^2}{\Delta r^2}-\frac{(R+j\Delta r)\Delta\theta^2}{\Delta r}\right)h_{i,j-1}-2h_{i-1,j}=0 \qquad (4.134)$$

and:

$$h_{i,j}=\frac{2(R+j\Delta r)^2\Delta\theta^2+(R+j\Delta r)\Delta\theta^2\Delta r}{4(R+j\Delta r)^2\Delta\theta^2+4\Delta r^2}h_{i,j+1}+\frac{2\Delta r^2}{4(R+j\Delta r)^2\Delta\theta^2+4\Delta r^2}h_{i+1,j}+$$
$$\frac{2(R+j\Delta r)^2\Delta\theta^2-(R+j\Delta r)\Delta\theta^2\Delta r}{4(R+j\Delta r)^2\Delta\theta^2+4\Delta r^2}h_{i,j-1}+\frac{2\Delta r^2}{4(R+j\Delta r)^2\Delta\theta^2+4\Delta r^2}2h_{i-1,j}=0 \qquad (4.135)$$

Where: i = 0, 1, ...m nodes in θ direction, j = 0, 1, ...n nodes in r direction; Δr and Δθ are the radial and the tangential step length, respectively.

The Stokes equations for the flow in Ω_2 sub-domain are:

$$\nabla.u=0$$
$$-\mu\Delta u+\nabla p=0 \qquad (4.136)$$

The pressure boundary integral of equations (4.136) is given by:

$$h(r,\theta)=-\frac{2\mu}{r}\{[\cos\theta(r\frac{\partial}{\partial r}P(r,\theta))-\sin\theta\frac{\partial}{\partial\theta}P(r,\theta)]*u_r(R,\theta)$$
$$+[\sin(\theta)(r\frac{\partial}{\partial r}P(r,\theta))+\cos\theta\frac{\partial}{\partial\theta}P(r,\theta)]*u_\theta(R,\theta)\}+\frac{1}{2\pi}\int_0^{2\pi}h(R,\theta)d\theta \qquad (4.137)$$

Where: $P(r,\theta)=\dfrac{R^2-r^2}{2\pi[R^2+r^2-2rR\cos\theta]}$

The coupling method algorithm:

Set initial guess for the pressure λ^0,

For s =0, 1, 2, ... continue until convergence,

1. Apply the following boundary condition on the interface Γ and Γ_1, respectively:

$$h_{i,0} = \lambda^0, i = 0,1,.....m$$
$$h_{i,n} = h_e - (h_e - h_w)\frac{R_1 + R_1\cos(i.\Delta\theta)}{2R_1} \quad (4.138)$$

2. Solve equation (4.134) for pressure $h^s = \lambda^s$ in FD sub-domain,

3. Obtain the nodal velocities at the interface Γ by:

$$u_r^s = -\frac{K}{\mu}\frac{\partial h^s}{\partial n}$$
$$u_\theta^s = -\frac{K}{\mu}\frac{1}{r}\frac{\partial h^s}{\partial \theta} \quad (4.139)$$

4. Solve equation (4.139) for pressure λ^{s+1} in BE sub-domain using the velocities obtained in step 3,

5. Apply:

$$\lambda^s = \lambda^{s+1} \quad (4.140)$$

6. Repeat the above steps until convergence.

He et al. [118] applied an iterative DDM technique to solve the conjugate heat transfer with an incompressible flow problem within a thick-walled parallel-plate channel using a coupled FD/BE method. The incompressible convection heat transfer equation in the fluid sub-domain (a liquid film) is discretized using the FDM, and the conduction heat transfer equation in the solid sub-domain (a parallel-plate channel) is approximated using the BEM, see Figure 4.4.

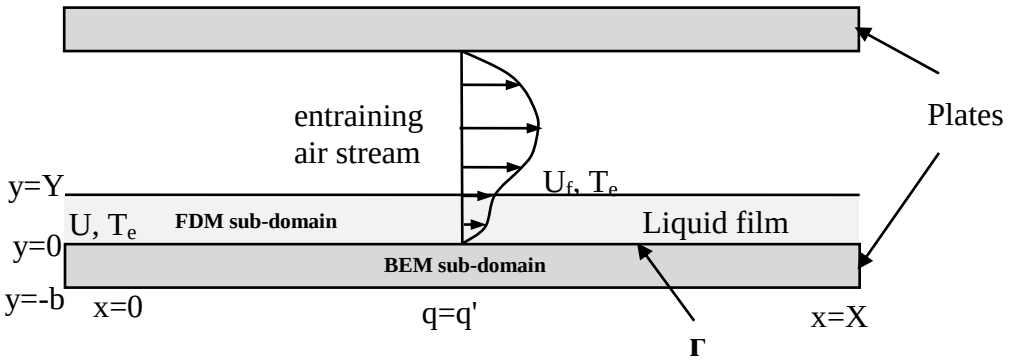


Figure 4.4 FDM and BEM sub-domains

The steady convection heat transfer equation, which governs the flow in the FDM sub-domain, is given by:

$$\rho c_p\left(u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}\right)=k_f\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)+\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2 \tag{4.141}$$

Where: ρ is the fluid density, c_p is the specific heat, T is the temperature, k_f is the fluid conductivity, μ is the fluid viscosity, u and v=0 are the fluid velocity components.

He et al. used a first-order and a second-order central finite difference formulations to compute the convective derivative and diffusion terms in equation (4.141), respectively.

Laplace equation for the BEM sub-domain, seen in equation (4.97), is rewritten using the reference [118] symbols as:

$$\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)=0 \tag{4.142}$$

The boundary integral equation (see equation 4.101) for the BEM sub-domain is:

$$C(P)T(P)=\int_\Gamma T^*(P,Q)\frac{\partial T(Q)}{\partial n}d\Gamma-\int_\Gamma \frac{\partial T^*(P,Q)}{\partial n}T(Q)d\Gamma \tag{4.143}$$

Where: $T^*(P,Q)$ is the Green function for Laplace equation, see equation (4.102).

The discretized form of equation (4.143) is given by:

$$[\mathbf{H}]\{\mathbf{T}_B\} = [\mathbf{G}]\{\mathbf{q}''_B\} \tag{4.144}$$

Where: $\{\mathbf{T}_B\}$ and $\{\mathbf{q}''_B\}$ are the solid nodal temperature and heat flux vectors, respectively.

The method sequential Neumann- Dirichlet coupling algorithm is:

For n =0, 1, 2, ... continue/repeat until convergence,

1. Set initial guess for $\{\mathbf{q}''^{I}_{B,0}\}$,

2. Obtain $\{\mathbf{T}^{I}_{B,n}\}$ in the BE sub-domain by solving equation (4.144),

3. Apply the continuity conditions: $\{\mathbf{T}^{I}_{F,n}\} = \{\mathbf{T}^{I}_{B,n}\}$ (4.145)

4. Obtain $\{\mathbf{q}''^{I}_{F,n}\}$ in the FD sub-domain by solving the discretized form of equation (4.141),

5. Check the solution convergence if:

$$\left\|\mathbf{q}''^{I}_{F,n} - \mathbf{q}''^{I}_{B,n}\right\| \le \varepsilon \tag{4.146}$$

Stop if the convergence condition is satisfied; otherwise proceed to the next step,

6. Apply:

$$\{\mathbf{q}''^{I}_{B,n+1}\} = \frac{\omega_B\{\mathbf{q}''^{I}_{B,n}\} + \omega_F\{\mathbf{q}''^{I}_{F,n}\}}{\omega_B + \omega_F}, \text{ for } n \ge 1 \tag{4.147}$$

Where: ω_B and ω_F are weighting parameters.

7. Return to step (2) and repeat the steps that follow until convergence.

Ingber et al. [119] developed a domain decomposition method to solve the 3-D diffusion problems, governed by Helmholtz equation using the parallel Neumann-Neumann iteration scheme proposed by Kamiya et al. [82] (the scheme is presented earlier in this chapter). The 3-D domain Ω is divided into multiple sub-domains, and the PDE in each sub-domain is solved

using BEM and discretization in time. The diffusion problems Ingber et al. solved are governed by the following equation:

$$\rho c_p \frac{\partial u}{\partial t} + g = k\nabla^2 u \tag{4.148}$$

Where: u is the temperature and g is an internal heat generation.

The displacement time derivative appears in equation (4.148) is discretized using an FDM, called the generalized trapezoidal θ method. This equation becomes:

$$\nabla^2 v^n = \frac{v^n}{\theta\alpha\Delta t} - \frac{u^{n-1}}{\theta^2\alpha\Delta t} + \frac{(1-\theta)g^{n-1}}{\theta k} + \frac{g^n}{k} \tag{4.149}$$

Where: The diffusivity $\alpha = k/\rho c_p$, Δt is the time step, superscript n is time step number, and v^n is the transformed temperature at time step n:

$$v^n = u^n - \frac{\theta-1}{\theta} u^{n-1} \tag{4.150}$$

The boundary integral representation of v^n in equation (4.149) is given by:

$$\begin{aligned} \eta(P)v^n(P) &= \int_\Gamma [\frac{\partial G(P,Q)}{\partial n} v^n(Q) - G(P,Q)\frac{\partial v^n(Q)}{\partial n}]d\Gamma(Q) \\ &+ \int_\Omega \left[\frac{v^n(Q)}{\theta\alpha\Delta t} - \frac{1}{\theta^2\alpha\Delta t} u^{n-1}(Q) + \frac{1-\theta}{\theta k} g^{n-1}(Q) + \frac{g^n(Q)}{k} \right] G(P,Q)d\Omega(Q) \end{aligned} \tag{4.151}$$

Where: Green function G(P,Q) = 1/r and $\eta(P) = \int_\Gamma \frac{\partial G(P,Q)}{\partial n} d\Gamma(Q)$

The discretized forms of equation (4.149) and the integral in equation (4.151), written in vector and matrix form, are given by the following two systems of equations:

$$[A_{ij}]\{v_j^n\} - [A_{ij}][\Phi_{jk}]\{\alpha_k^n\} = [B_{ij}]\{q_j^n\} - [B_{ij}][\Phi'_{jk}]\{\alpha_k^n\} \tag{4.152}$$

$$\begin{aligned} &\alpha\theta\Delta t[\psi_{mi}]\{\alpha_i^n\} - [\tilde{\Phi}_{mi}]\{\alpha_i^n\} + \frac{1}{\theta}\{u_m^{n-1}\} - \frac{\alpha\theta\Delta t}{k}\left[\{g_m^n\} + \frac{1-\theta}{\theta}\{g_m^{n-1}\}\right] \\ &+ [C_{mj}]\{v_j^n\} - [C_{mj}][\Phi_{jk}]\{\alpha_k^n\} = [D_{mj}]\{q_j^n\} - [D_{mj}][\Phi'_{jk}]\{\alpha_k^n\} \end{aligned} \tag{4.153}$$

The system of equations (4.152) and (4.153) are solved together for the unknowns $\{v_j^n\}$ and $\{\alpha_k^n\}$.

Where: α_k^n is *k*th the radial basis function coefficient at time step n; Φ_{ji} and Φ'_{ji} are the value of ϕ_i at jth boundary node and its normal derivative, respectively; ϕ_i is the particular solution associated with the radial basis function, ψ; ψ_i is *i*th radial basis function; $[\psi_{mi}]$ is the matrix to determine the radial basis function coefficient; $[A_{ij}],[B_{ij}],[C_{ij}]$ and $[D_{ij}]$ are the boundary element coefficient matrices.

The method parallel Neumann-Neumann coupling algorithm is:

Set initial guess for all interfacial fluxes zero values, i.e.

$$\{q_0^{i,0}\} = 0 \text{ and } \{q_0^{j,0}\} = 0$$

Where: $\{q_0^{i,0}\}$ and $\{q_0^{j,0}\}$ are the nodal fluxes on the interface between sub-domain (i) and sub-domain (j) at the initial time step (the second superscript) and at the initial iteration (the subscript).

For n =0, 1, 2, ... specified number of time steps,

1. For m =0, 1, 2, ... continue until convergence,

1.1. Obtain $\{v_m^{i,n}\}$ and $\{\alpha_m^{i,n}\}$ by solving equations (4.152) and (4.153) in sub-domain i,

1.2. Obtain $\{v_m^{j,n}\}$ and $\{\alpha_m^{j,n}\}$ by solving equations (4.152) and (4.153) in sub-domain j,

1.3. Apply:

$$\{q_m^{i,n}\} = \{q_{m-1}^{i,n}\} + \gamma(\{v_{m-1}^{i,n}\} - \{v_{m-1}^{j,n}\}) \tag{4.154}$$

and

$$\{q_m^{i,n}\} = \{q_m^{j,n}\} \tag{4.155}$$

Where: γ is a relaxation parameter.

1.4. Check the solution convergence:

$$\max\left\|v_m^{i,n} - v_m^{j,n}\right\| \leq \varepsilon \tag{4.156}$$

1.5. Repeat the steps from 1.1 to 1.4 until convergence.

2. Apply in the next time step, n > 0:

$$\{q_0^{i,n}\} = \{q_0^{i,n-1}\} \quad \text{and} \quad \{q_0^{j,n}\} = \{q_0^{j,n-1}\} \tag{4.157}$$

Chapter 5

The Developed BEM-FDM (FLAC^{3D}) Coupling Methods

/Numerical Examples and Applications/

5.1 Introduction

The main objectives for coupling BEM with FDM in this thesis are combining the advantages and avoiding the disadvantages of both methods, validating the developed coupled FEM or FDM with BEM (s) in the existing research works, and extending the capabilities of a well-known commercial FDM program (FLAC^{3D}) in solving elasto-plastic problems in continuous three-dimensional infinite and semi-infinite media.

Truncation Boundaries

Generally, users should take into consideration two types of solution convergence while solving a problem numerically using Flac^{3D}. The first type is in terms of time steps: the solution converges after a number of time steps if either equilibrium or steady state is reached; otherwise the model is not in an equilibrium state. The second type of convergence is in terms of zone size or total number of zones of the discretized medium. As shown before in Chapter 3, because Flac^{3D} uses a constant strain-rate zone (constant stress zone), the solution converges at a higher cost in terms of number of zones than that in the other programs where higher order elements can be used. The solution converges, especially in high stress gradient regions, when finer meshes are used. The zone aspect ratio should almost be equal to one, and a gradual change of the zones' size should be used when required [36].

Whether the modelled medium is infinite or semi-infinite another type of convergence should be taken into consideration. This type of convergence is in terms of truncation boundaries location. Because no computer memory can cover the whole body with zones, artificial

(truncation) boundaries must be introduced. Having this extra type of convergence requires more effort and more run and analysis time to reach the desired accuracy of the solution. If the truncation boundaries are placed very far from the area of interest, the solution will converge asymptotically to the exact solution, but the farther the truncation boundaries are located, more zones are needed for modelling and the runtime increases.

> The solution time for a Flac^{3D} run is proportional to $N^{4/3}$, where N is the number of zones. This formula holds for elastic problems, solved for the equilibrium condition. The runtime will vary somewhat, but not substantially, for plasticity problems, and it may be much larger if continuing plastic flow occurs. [37]

Introducing artificial boundaries also introduces two types of error. Those errors are as quoted from the Flac^{3D} manual:

"(1) A fixed boundary causes both stresses and displacements to be underestimated.

(2) A stress boundary causes both stresses and displacements to be overestimated." [38]

Another source of error is introduced because of the boundary location as the manual warns "If plastic flow occurs along such a boundary (artificial boundary), then the solution is not realistic, because the mechanism of failure is influenced by a nonphysical entity" [39]. Flac^{3D} manual recommends the following to reduce the errors: "Artificial boundaries are placed sufficiently far away from the area of interest that the behaviour in that area is not greatly affected" [40]. That requires, as explained before, more Flac^{3D} runtime and more user effort and analysis time to achieve.

The BEM inherently satisfies the differential equations in infinite and semi-infinite domains and FEM/FDM better approximates the problems' solutions in the finite region of geometrical and/or material non-linearity. If the bounded region which includes the area of interest (eg.

mining field, excavated opening, buried structure with part of the surrounding rock or soil strata) is analyzed numerically by FLAC^{3D} knowing that "Linear simulations run more slowly with FLAC^{3D} than with equivalent finite element programs. FLAC^{3D} is most effective when applied to nonlinear or large-strain problems, or to situations in which physical instability may occur" [31]. On the other hand, if the remaining infinite/semi-infinite linear elastic region is analyzed by the BEM, the following will be accomplished:

- ❖ No further discretization to the unbounded region is required.
- ❖ Faster solution convergence is achieved with less running and analysis time.
- ❖ The computed mechanical responses by Flac^{3D} are corrected, and the above-mentioned sources of error are eliminated.
- ❖ The mechanical responses very far from Flac^{3D} bounded domain (e.g. at the ground surface) in the surrounding infinite or semi-infinite domain are computed with less cost in runtime or in the required number of finite difference constant strain rate elements.

5.2 Proposed Iterative BEM-FDM Coupling Algorithms

Simple forward, backward, and central difference formulations are used mainly to couple the FDM with the BEM in the existing research [93-120]. These research works succeeded in the last two decades in coupling the FDM with the BEM to solve diffusion time independent [45], [97], [98], [117], [118] and time dependent [5], [99-103], [119], [120] problems with simple geometry and to solve some other time-dependent problems such as dynamics [111-114], and acoustics [112]. Although the time-independent simple geometry elasto-plasticity problems were solved in the seventies [93-95] using coupled FD with the BE method, the recent extensively researched coupled FE with the BE methods [3], [46], [47], [51], [57], [56], [61-65], [68], [70-90] surpassed the coupled FD with the BE method in solving these types of

problems and others with more complicated geometry and/or mechanical behaviour nonlinearity. Flac^{3D} as described in its manual is "an explicit finite difference program," [29] but it shares the following characteristics with FEM:

> Both methods translate a set of differential equations into matrix equations for each element, relating forces at nodes to displacements at nodes. Although FLAC^{3D}'s equations are derived by the finite difference method, the resulting element matrices for an elastic material are identical to those of the finite element method (for constant-strain tetrahedra). [31]

Consequently, the already developed coupled FEM/BEM techniques are applicable in the proposed coupled FD (FLAC^{3D}) method with the BE method.

The existing coupled FE-BE methods achieved the coupling either at the level of discretized equations [3], [46], [47], [56], [61-65], or by using iterative domain decomposition methods [51], [57], [68], [70-90]. The first method requires assembling a complicated unified system of equations, contrary to the uncoupled BEM or FEM, which builds a simple separate system of equations for each single method [41]. Coupling in this method was achieved using two approaches, each approach has its advantages and disadvantages as detailed in section (4.2.1). The coupled FE-BE methods using iterative domain decomposition algorithms enjoys many advantages, as elaborated before in section (4.2.2), over the coupling methods at the level of discretized equations. The above reasons motivate the author to apply the iterative DDMs techniques in coupling FDM (FLAC^{3D}) with BEM. The DDM, which partitions the task of solving partial differential equations (PDE) numerically by splitting the original problem of a large and/or complex domain into a set of sub-domains [70], was employed to couple FEM with BEM [51], [57], [68], [70-90] and to couple FDM with BEM [98], [116-118] non-overlapping sub-domains using relaxation operators on the interfacing boundaries. Coupling BEM with

FEM using iterative domain decomposition techniques was studied relatively intensively in the last two decades as discussed before in section (4.2.2). Conversely, coupling BEM with FDM has developed recently to solve mainly case-specific diffusion problems with simple geometry.

Perera et al. [80], [81] developed an iterative domain decomposition FEM-BEM coupling method using the Steklov-Poincaré operator, which consists of two parallel Dirichlet-Nuemann coupling steps, as explained earlier in section (4.2.2). This method and the two parallel Neumann-Neumann and Dirichlet-Neumann FEM-BEM iterative coupling methods, developed by Kamiya and Iwase [82], have a specific disadvantage: If Neumann boundary conditions are imposed on the whole boundary either for the FEM sub-domain or the BEM sub-domain, a non-unique solution will result [41]. Gerstle et al. [68] developed a non-overlapping domain decomposition FEM-BEM coupling method using an iterative conjugate gradient solver based on Bjorstad and Widlund's [79] techniques. A trial displacement vector is applied on the FEM and BEM sub-domains' interface. The displacements are corrected iteratively by the solver, which uses the unbalanced forces on the interface to predict the new trial interfacial displacements for the next iteration. Kamiya and Iwase [83] replaced the parallel Neumann-Neumann and Dirichlet-Neumann methods they used before with the conjugate gradient method to renew the unknown integrated flux at the interface between the FEM and BEM sub-domains by utilizing the condense method. In this method only the unknowns on the interface are treated during the renewal iteration. The iterative methods, which use the conjugate gradient solver, are only applicable to the special case of symmetric BEM. Dong [88] employed the symmetrization-iteration method (SIM) in developing an iterative BE-FE coupling method. In this method, the condensation technique is used to reduce the BE sub-domain's stiffness matrix. The convergence of Dong's method depends on the choice of an enhanced symmetric matrix,

which demands an access into the FEM sub-domain stiffness matrix. Lin et al. [51] developed an iterative FE/BE coupling method using a sequential Dirichlet-Neumann scheme. This method employed the spectral collocation approximation, and an interface relaxation approach with an automatic selection of the relaxation parameter at each iteration. The algorithm, as detailed in section (4.2.2), first imposes an initial nodal displacement vector on the interface for the BEM sub-domain. Second, the solved nodal traction vector is applied, and converted into nodal forces vector, according to the equilibrium condition on the interface as a BC for the FEM sub-domain in the next step. Third, the solved and the assumed interfacial displacement vectors are checked for both domains in order to satisfy the convergence condition; otherwise, the iteration continues until convergence is achieved. Feng and Owen [57] employed a similar sequential Dirichlet-Neumann scheme to develop an iterative FE/BE coupling to find the solution for a plate resting on an elastic half space foundation problem. The BEM and the FEM are used to analyze the elastic homogeneous half space medium, and the nonlinear inhomogeneous plate behaviours, respectively. The method convergence was investigated by Elleithy et al. [41], who established two conditions (equations 4.86 & 4.87); the iterative method should satisfy to achieve the convergence, see section (4.2.2). Although El-Gebeily et al. [124] established the convergence for the parallel Dirichlet-Neumann and Neumann-Neumann domain decomposition FE/BE coupling methods, the convergence conditions and the selection of the relaxation parameters are still in need for more research until they become well-established. Elleithy and Tanaka [70], [78] studied the convergence of the geometric contraction based, Robin relaxation, and Dirichlet/Neumann averaging methods. However, establishing the convergence conditions must remain the subject of future research. In contrast, Elleithy et al. [41] well defined the convergence conditions, whereas, the relaxation parameter

selection, is restricted to the condition's limits. Therefore, the sequential Dirichlet-Neumann scheme is utilized to develop an iterative FD ($FLAC^{3D}$)-BE domain decomposition coupling method in this thesis.

The elasto-plastic mechanical response in the FD bounded sub-domain is approximated numerically by the $FLAC^{3D}$ program. The program is very effective in solving nonlinear or large-strain problems, as mentioned before. On the other hand, the linear elastic response of the surrounding infinite or semi-infinite sub-domain is computed using the BEM. Coupling both sub-domains would combine the advantages of both methods (see these advantages in section 5.2). No further discretization to the unbounded region is required. Hence, the sub-domains' interface is already discretized into the number of first order quadrilateral boundary elements. These BEs represent the base of the $Flac^{3D}$ zones alongside the interface. The algorithm of the proposed sequential Dirichlet-Neumann iterative domain decomposition method (IDDM), utilizing Perera et al. [81] and Elleithy and Tanaka [70] methods, to couple the nonlinear plastic behaviour of FDM sub-domain with the linear elastic one of BEM sub-domain is:

For n =0, 1, 2,... continue until convergence,

1. Set initial guess for $\{\mathbf{u}^{I}_{F,0}\}$,
2. Obtain $\{\mathbf{f}^{I}_{F,n}\}$ in the FD sub-domain using $Flac^{3D}$ program. The program computes the nodal forces as outlined in the following steps (see section 3.5 for more details):

   For i = 1, 2, ... Total number of steps*,

   1. Compute the strain rate tensor for each constant strain-rate tetrahedron by:

$$\xi_{ij} = -\frac{1}{6V}\sum_{l=1}^{4}\left(\upsilon_i^l n_j^{(l)} + \upsilon_j^l n_i^{(l)}\right)\cdot S^{(l)}$$

2. According to the second and third approaches of numerical formulations section, the medium mass is discretized and redistributed by computing the tetrahedron contribution of the nodal mass by using the following equation:

$$m^{l} = \frac{\alpha_1}{9V^{[l]}} \max\left(\left[n_i^{(l)} S^{(l)} \right]^2, i = 1,3 \right)$$

Where: $\alpha_1 = K + 4/3G$, K is bulk modulus and G is shear modulus.

3. Obtain the tetrahedron contribution at the node, locally numbered as *l*, by the equation:

$$p_i^{l} = \frac{1}{3}\sigma_{ij} n_j^{(l)} S^{(l)} + \frac{1}{4}\rho b_i V$$

4. Sum up the contributions of all the tetrahedra that share the node globally numbered as < *l* >. This summation operation is given the symbol $[[p_i]]^{<l>}$.

5. Compute the out of balance force as the difference between the external applied force $P_i^{<l>}$ at node < *l* > and the internal force $[[p_i]]^{<l>}$ as:

$$F_i^{<l>} = [[p_i]]^{<l>} + P_i^{<l>}$$

6. Compute mechanical damping force (will be added to the out of balance force in the next step) by the following equation:

$$\boldsymbol{F}_i^{<l>} = -\alpha . \left| F_i^{<l>} \right| . \mathrm{sign}\left(\upsilon_i^{<l>} \right)$$

Where: $\mathrm{sign}\left(\upsilon_i^{<l>} \right) = \begin{cases} +1, & \text{if } : \upsilon_i^{<l>} > 0; \\ -1, & \text{if } : \upsilon_i^{<l>} < 0; \\ 0, & \text{if } : \upsilon_i^{<l>} = 0 \end{cases}$

α is a constant given by Flac^{3D} the value of 0.8.

7. Compute new nodal velocity using central finite difference formulation in time and equations (3.39) and (3.42) as:

$$\upsilon_i^{<l>}\left(t+\frac{\Delta t}{2}\right)=\upsilon_i^{<l>}\left(t-\frac{\Delta t}{2}\right)+\frac{\Delta t}{M^{<l>}}\left(F_i^{<l>}+\boldsymbol{F}_i^{<l>}\right)$$

8. Compute nodal displacement by:

$$u_i^{<l>}(t+\Delta t)=u_i^{<l>}(t)+\Delta t.\upsilon_i^{<l>}\left(t+\frac{\Delta t}{2}\right)$$

9. Update the geometry if the large strain mode is chosen by the user using the equation:

$$x_i^{<l>}(t+\Delta t)=x_i^{<l>}(t)+\Delta t.\upsilon_i^{<l>}\left(t+\frac{\Delta t}{2}\right)$$

If the small strain mode is used, which is the default mode, no geometry update is done.

10. Repeat steps 3 to 10 until state of equilibrium or steady flow is reached; otherwise, indicate state of non-equilibrium.

3. Obtain $\{\mathbf{t}_{B,n}^{I}\}$ according to equation $\{\mathbf{f}_{F,n}^{I}\}+[\mathbf{M}]\{\mathbf{t}_{B,n}^{I}\}=0$,

4. Obtain $\{\mathbf{u}_{B,n}^{I}\}$ in the BE sub-domain by solving equation (4.26),

5. Apply eq. (4.83):

$\{\mathbf{u}_{F,n+1}^{I}\}=\omega\{\mathbf{u}_{B,n}^{I}\}+(1-\omega)\{\mathbf{u}_{F,n}^{I}\}$, for n ≥ 1

Where: ω is a relaxation parameter.

6. Check the solution convergence:

$$\left\|\frac{\mathbf{u}_{F,n+1}^{I}-\mathbf{u}_{F,n}^{I}}{\mathbf{u}_{F,n+1}^{I}}\right\|\leq\varepsilon \qquad (5.32)$$

If the convergence condition is satisfied, stop; otherwise, proceed to the next step,

7. Return to step (2) and repeat the steps that follow until convergence.

Two main points are taken in consideration in the algorithm. First, although the initial guess for the interfacial displacement may take any value, as Lin et al. [51] suggested, it started with zero value. This assumption, which seems physically reasonable, corresponds to the case of fixed interface. Second, the total step number (*p.131 and p. 269) the Flac^{3D} program needs to obtain the nodal interfacial displacement $\{\mathbf{f}_{F,n}^{I}\}$, in the stress boundary condition case, is extracted initially before the algorithm's iteration starts. The non-equilibrium case is not studied in this thesis (see the algorithm flow charts in Figure 5.a in Appendix a, p.289).

5.3 Numerical Examples

Five examples are solved: the first is a spherical excavation in a three dimensional infinite medium; the second is a square uniform load applied on the infinite plane (z =0) of a 3D semi-infinite space; the third is a cylindrical tunnel in an infinite Medium (2D plane strain problem), the fourth is a hole near the edge of a semi-infinite plate under tension (2D plane stress problem); and the fifth is a smooth square footing on a cohesive frictionless material in a 3D semi-infinite medium.

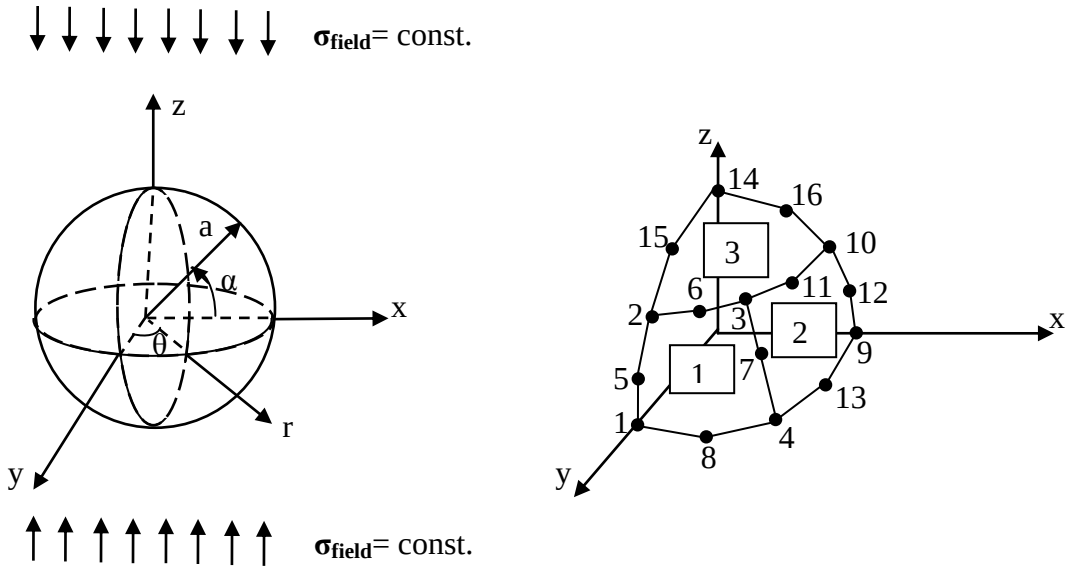
5.3.1 Spherical Cavity in 3D Infinite Medium

This problem was solved theoretically by R. V. Southwell in 1926 and presented by Timoshenko and Goodier [125] who derived the stress distribution around a spherical cavity in an infinite medium subjected to a uniform tension/compression in one direction (see Figure 5.1). In Geo mechanics the cavity is similar to a spherical excavation problem. The medium, subjected to a compressive stress field, can be considered as an infinite domain if the excavation is very deep. The field stress in this example [3], [47] is σ_z = -1 MPa and the other stress components are zero. E=1000 MPa, Poisson's ratio=0. Sphere radius a = 1 m. According to Cauchy rule $t_i = \sigma_{ji} n_j$ we get: $t_x = 0, t_y = 0, t_z = \sigma_{zz} n_z$; where : $n_z = \sin\alpha$

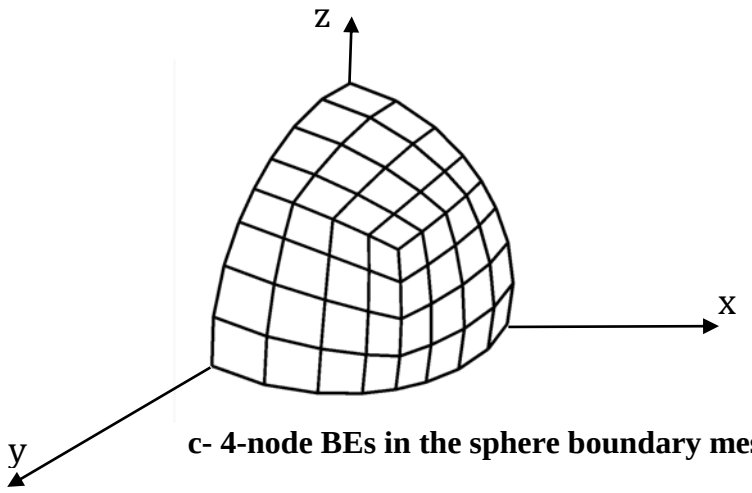
5.3.1.1 Uncoupled BEM Solution

The sphere boundary is discretized into 24 quadrilateral isoparametric quadratic 8- node elements. Because of the geometrical and loading symmetry about the three planes xy, xz, and yz, only 1/8 of the sphere is discretized. Total number of elements is three which have 16 nodes of three degrees of freedom each. Although this is a coarse mesh but a good agreement with the theoretical solution is obtained. The vertical displacement at the top of the sphere (Node 14 in Figure 5.1.b) is obtained numerically by solving the BIE (2.37). This equation is discretized into a system of equations (2.60) and solved, for a Neumann BCs, to get the displacement u_z = -0.91238 mm, compared to the theoretical value of -0.9 mm. The used fundamental solutions appear in equation (2.37) are 3D -Kelvin's fundamental displacement and traction solutions, respectively. The normalized stress (σ_z / σ_{field}) at internal points in the plane (z=0) is computed by solving the discretized form of equation (2.67) and plotted with the theoretical values in Figure 5.2, see table 5.1. Very good agreement between both methods is achieved as shown in the figure. Theoretical stress formula is [125]:

$$\sigma_z = \sigma_{field}\left[1+\frac{4-5\nu}{2(7-5\nu)}\frac{a^3}{r^3}+\frac{9}{2(7-5\nu)}\frac{a^5}{r^5}\right] \tag{5.1}$$



a. Spherical cavity in infinite medium **b. 8-Node BEs in the boundary mesh**



c- 4-node BEs in the sphere boundary mesh

Figure 5.1. Boundary elements (a, b, c) discretization for a spherical cavity boundary

The same problem is solved with finer mesh but with first order 48 BEs, see Figure 5.1.c. The obtained displacement at the top of the sphere was u_z= -0. 90675 mm and the normalized stress (see Table 5.1 and Figure 5.2) are with good agreement with the exact solution. The approximate solution was achieved with higher cost in terms of BEs number and runtime. This makes coupling FD ($Flac^{3D}$) sub-domain and infinite or semi-infinite BEM sub-domain more

costly, because the sub-domains' interface is discretized into number of first order quadrilateral boundary elements. These BEs represent the base of the first order Flac^{3D} zones adjacent to the interface.

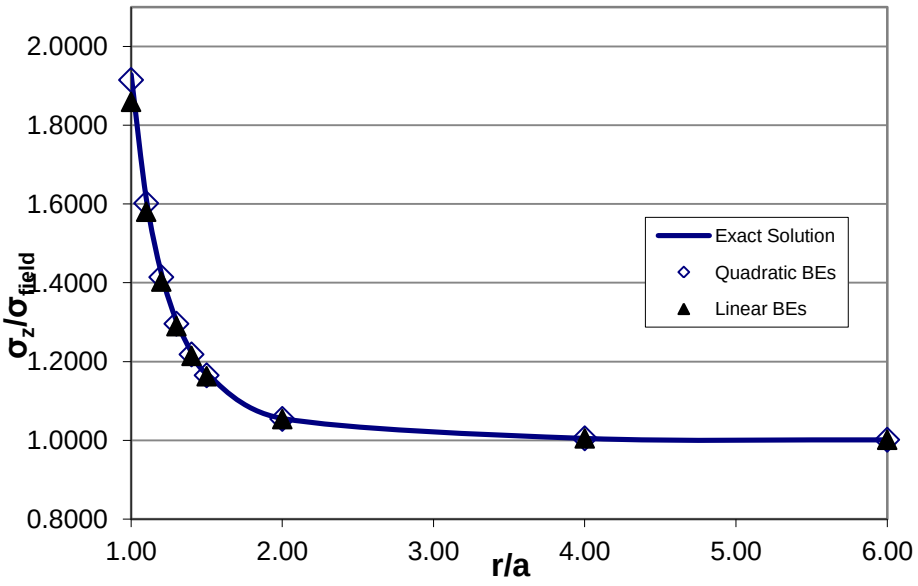


Figure 5.2. σ_z/σ_{field} distribution in the plane z = 0 in the radial direction

5.3.1.2 Uncoupled Flac^{3D} Solution

The same problem, a spherical cavity subjected to field stress (σ_z = -1 MPa) in the infinite medium, is solved again using Flac^{3D}. The solution's convergence in terms of truncation boundaries location is studied. The Flac^{3D} uncoupled bounded domain outer radius R is increased starting from 2 m, 3, 4, … up to 12 m with increment of 1 m, see Figure 5.3.

Table 5.1 Exact and numerical (BEM) normalized stresses σ_z/σ_{field} in plane (z = 0)

r/a	Quadratic BEs	Linear BEs	Exact Solution
1	1.9150	1.8596	1.9286
1.1	1.6020	1.5805	1.6138
1.2	1.4143	1.4031	1.4237
1.3	1.2960	1.2898	1.3032
1.4	1.2181	1.2145	1.2237
1.5	1.1650	1.1627	1.1693
2	1.0543	1.0538	1.0558
4	1.0050	1.0049	1.0051
6	1.0014	1.0014	1.0014
10	1.0003	1.0003	1.0003

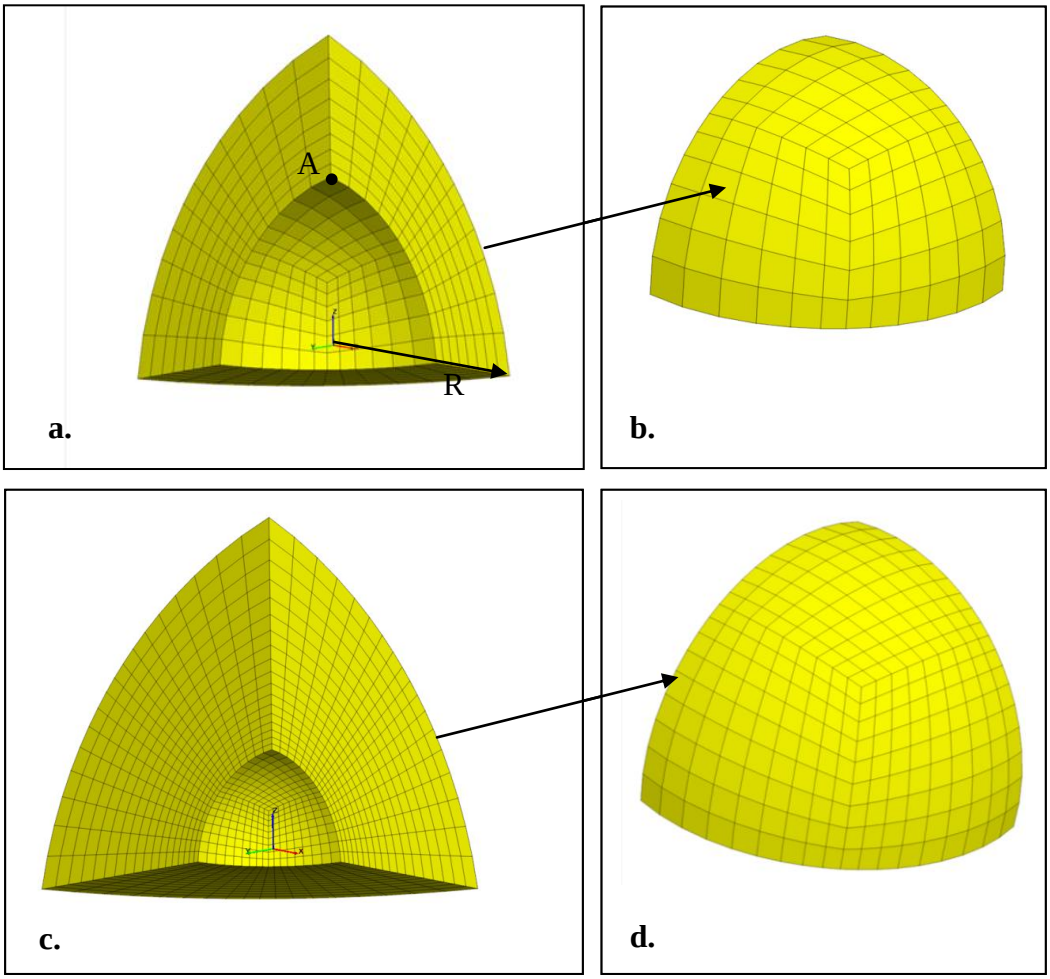


Figure 5.3. Flac3D Models of ratios: a. R/a = 2 and c. R/a = 4
b. and d. Flac3D-BEM sub-domains' interface

The vertical Displacement u_z at point A at the top of the spherical excavation is obtained numerically using Flac^{3D} program for each model of different outer radius (R). The vertical field stress (σ_z = -1 MPa) is applied on the truncation boundary making both stresses and displacements overestimated (see Figures 5.4 and 5.5 in Appendix b, p. 292). Hence, the displacement at the sphere top is over estimated in all models as a demonstration to what explained before. This displacement converges to the exact solution (0.9 mm) by increasing the Flac^{3D} model outer radius (R), as shown in Figure 5.6. The closest displacement to the exact solution is obtained numerically by a 12 m-outer-radius Flac^{3D} model, see Table 5.2. However,

better accuracy has been obtained using linear BEM with 3 quadratic or 48 linear BEs as shown before and with less cost in computer runtime. The Flac^{3D} model has 16025 nodes and discritized into 1440 zones and needs 31 seconds of computer runtime to solve. Conversely, the BEM model, which either has 14 nodes and discritized into three second order BEs or the one that has 61 nodes and discritized into 48 first order BEs, needs a second or less than a second of runtime for both BEM models to solve.

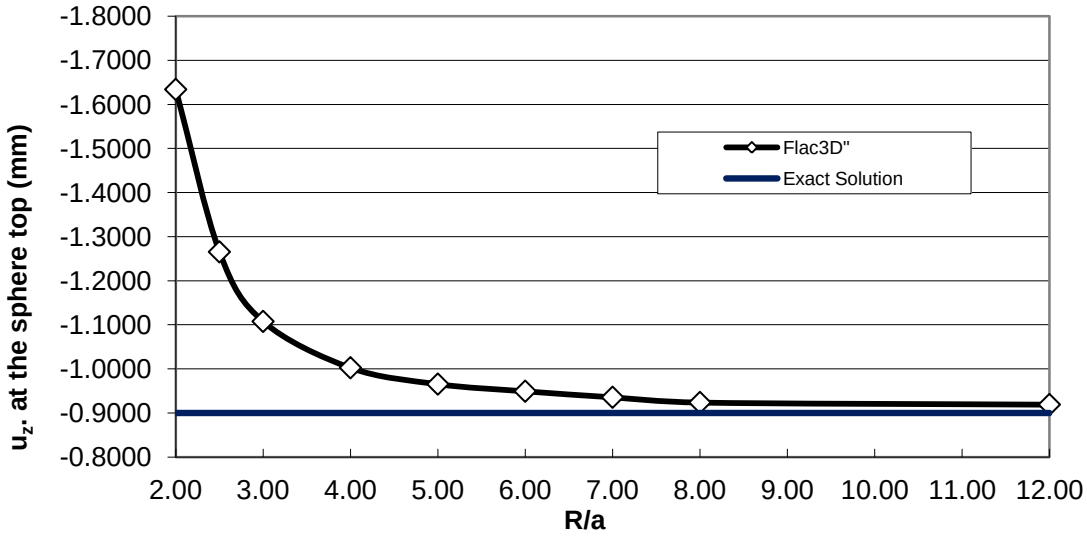


Figure 5.6 The development of u_Z as a function of R/a

Table 5.2 Exact and Flac3D vertical displacement u_Z at the sphere top

R/a	Falc3D (mm)	Exact Solution (mm)	Relative Error%
2	-1.6340	-0.9	81.55
2.5	-1.2654	-0.9	40.60
3	-1.1077	-0.9	23.07
4	-1.0025	-0.9	11.39
5	-0.9655	-0.9	7.28
6	-0.9493	-0.9	5.47
7	-0.9356	-0.9	4.51
8	-0.9237	-0.9	2.64
12	-0.9079	-0.9	2.11

The horizontal displacement u_x in plane (z=0) converges to the exact solution by increasing the Flac^{3D} bounded domain length. The solution converges faster starting from the domain of ratio R/a= 3, see Figure. 5.7.

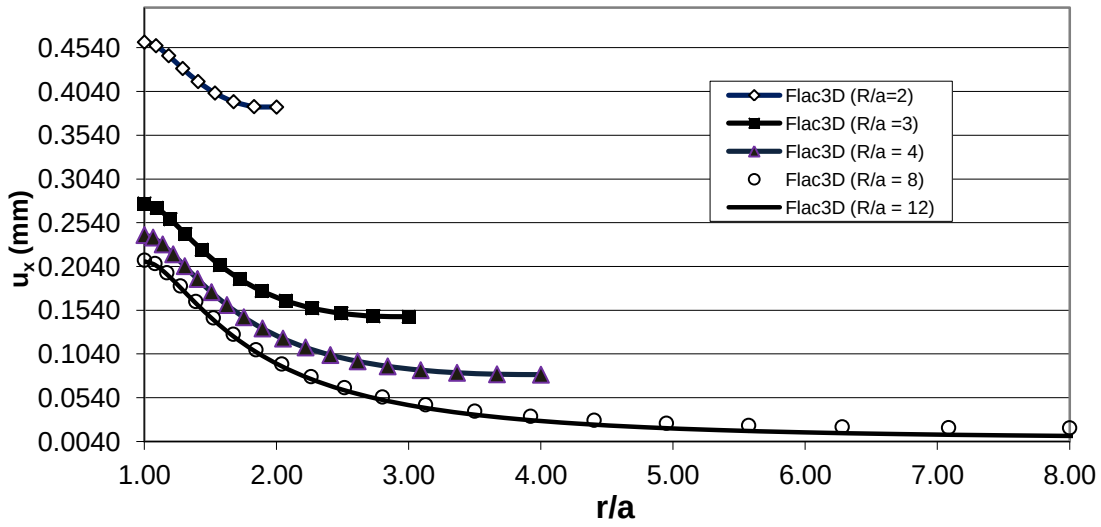
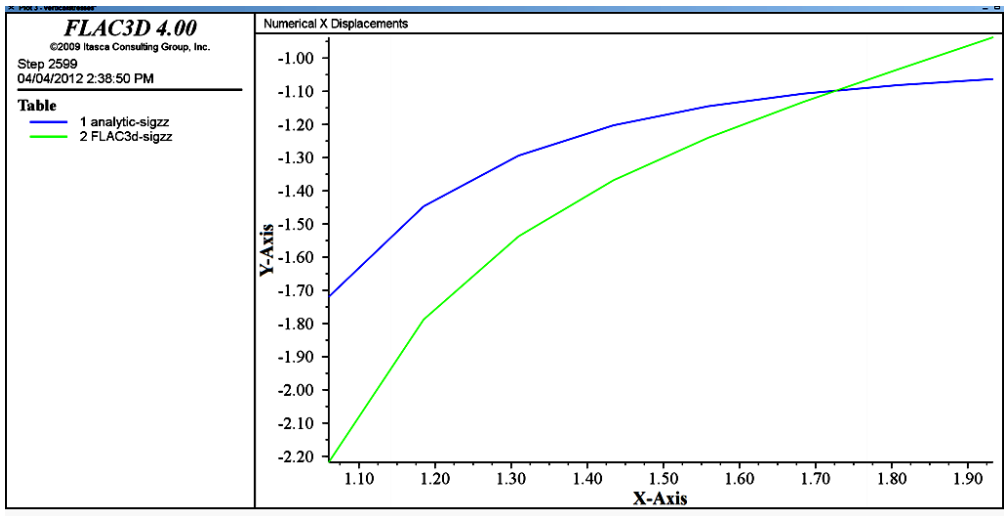


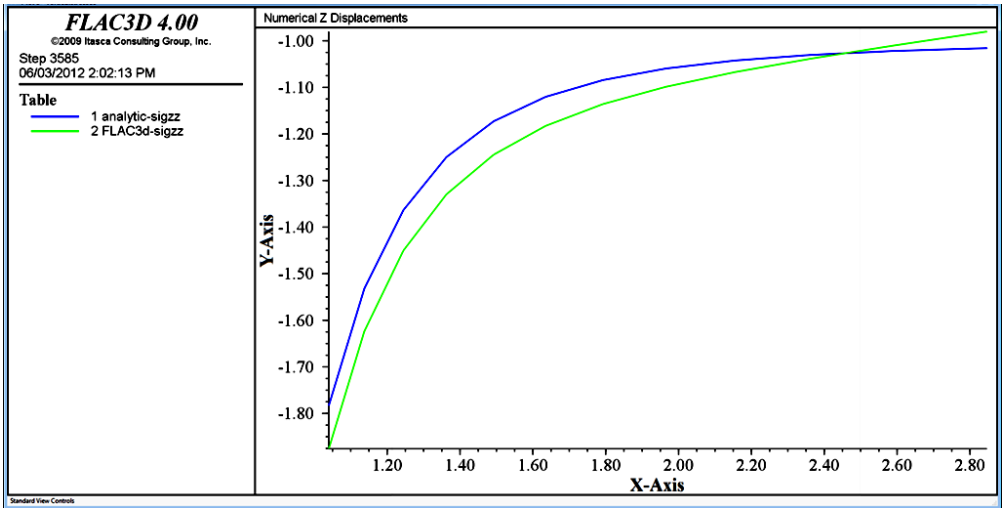
Figure 5.7 Development of u_x in plane (z = 0) as a function of R/a using Flac3D

The vertical stress σ_z converges to the exact solution faster than the displacement as shown in Figure 5.8 and Table 5.3 (see Appendix b, p. 293) . The numerical stress is also overestimated due to the same reason explained before.

a.



b.



c.

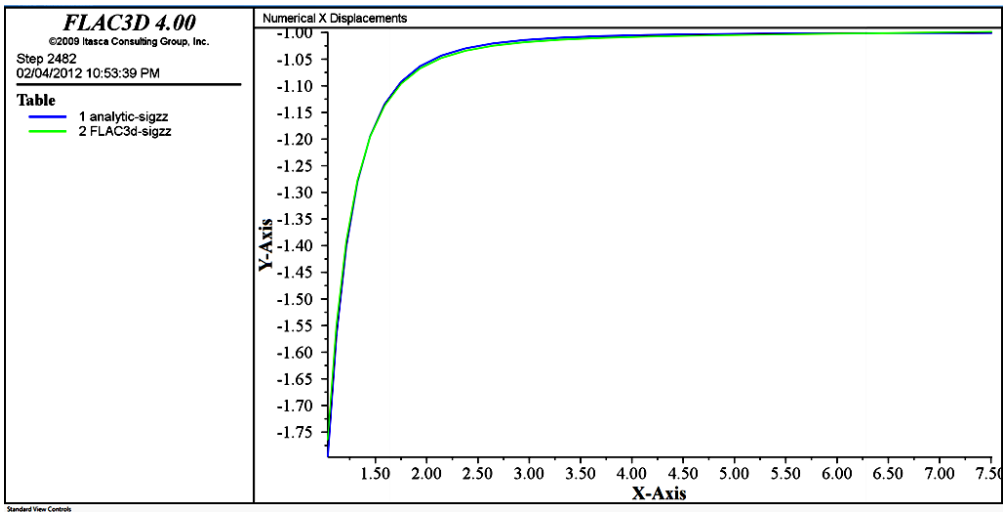


Figure 5.8 Exact and Flac3D vertical stress (σ_z) in plane (z = 0) as a function of (r) in the Flac3D models of the ratios: a. R/a = 2, b. R/a = 3 and c. R/a =8 , *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

5.3.1.3 Coupled Flac^{3D} -BEM Solution (First Method/IDDM)

The same problem is solved for the third time using the coupled FLac^{3D} -BE method with the algorithm detailed earlier in section (5.3). The Flac^{3D} model has the ratio R/a =2, see Figure 5.9. The computed mechanical responses has very good agreement with the theoretical solution compared to the uncoupled Flac^{3D} solutions with models of ratios R/a =8 and R/a =12, see Figure 5.9.

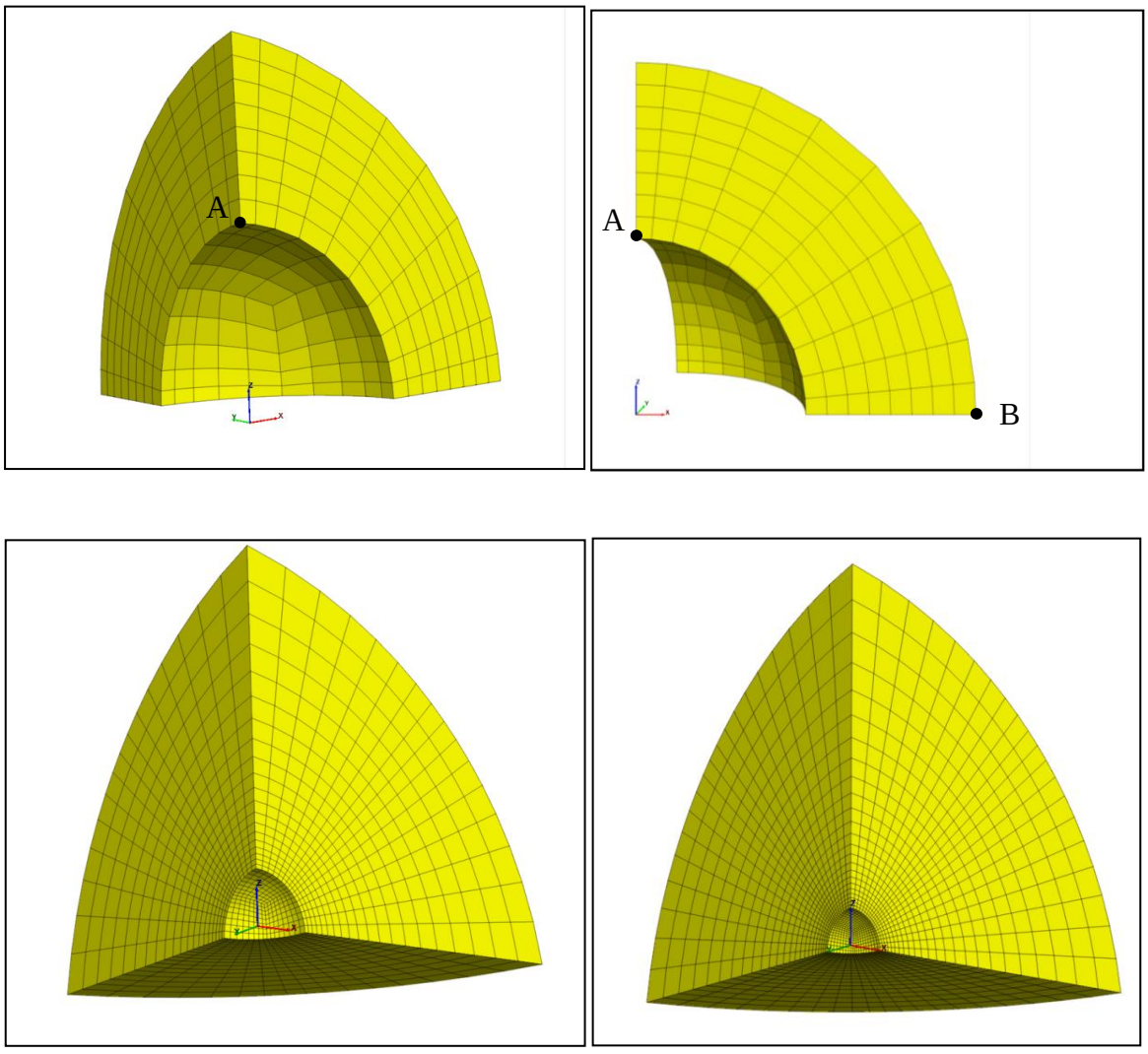


Figure 5.9. Flac3D Models of ratios: a. R/a = 2 b. R/a = 2
c. R/a = 8 d. R/a = 12

The computed vertical displacement value at point A using the coupled Flac^{3D} -BEM solution is corrected by more than 80 %, as shown in Figure 5.10 and Table 5.4. Seven iterations are needed in the coupled method with tolerance equals $\epsilon = 0.0014$ and relaxation parameter equals $\omega = 0.4$ to obtain the corrected solution. This value, as seen in Figure 5.10, lies between two soluions the first is an uncoupled solution with stressed interface (overestimated response solution) and the second is the zero-displacement- initial-guess solution (a fixed interface which produces an underestimated response solution) the iterations has started from. The corrected solution has the least relative error (R. E.), as shown in Table 5.4. It converges to the exact solution better than the uncoupled Flac^{3D} solutions with models of ratios R/a = 8 and 12, respectively. Although the coupled solution is a little more costly in terms of runtime (CPU) than the uncoupled ones but the mechanical responses are obtained in any point in the infinite BEM sub-domain with no significant extra cost.

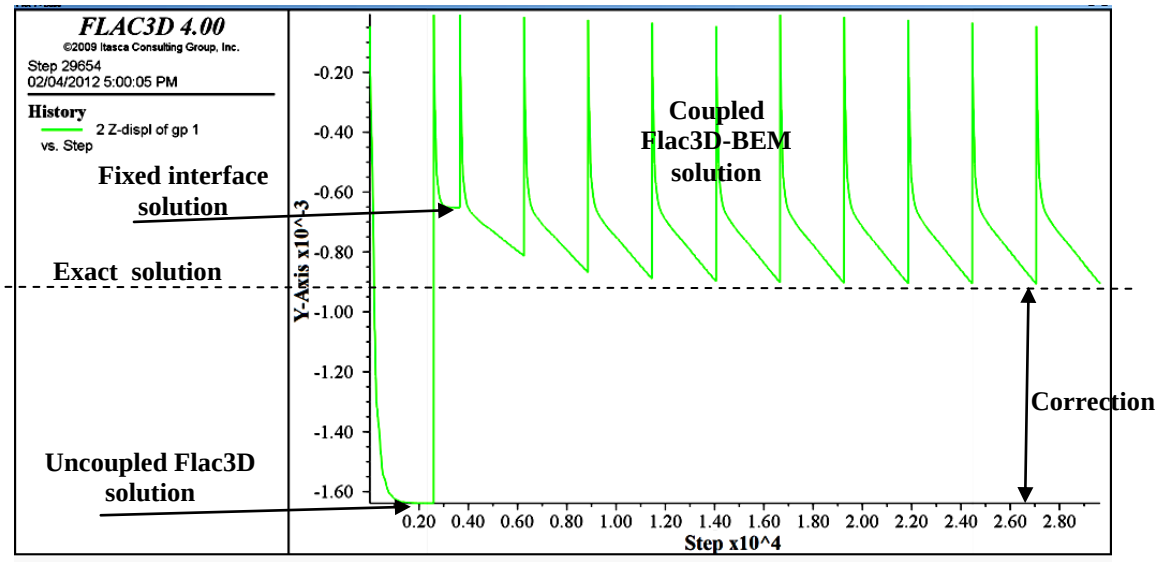


Figure 5.10 History of Flac3D and coupled vertical displacement at point A, *(Y-Axis ≡ u_Z).*

Table 5.4 Coupled Flac^{3D} -BEM and uncoupled Flac^{3D} vertical displacement values at point A

Method	R/a	Number of Zones	CPU [seconds]	u_z Numerical [mm]	u_z Exact [mm]	Relative Error%
Falc^{3D}	2	1200	3.81	-1.6340	-0.9	81.55
Falc^{3D}	8	7680	22.17	-0.9237	-0.9	2.64
Falc^{3D}	12	14400	31.37	-0.9190	-0.9	2.11
Falc^{3D} -BEM /First Method /	2	1200	38.795	-0.9065	-0.9	0.68
Falc^{3D} -BEM /Optimum Solution /	2	1200	28.719	-0.9062	-0.9	0.69

The vertical stress in plane (z =0) is also with very good agreement with the exact solution either in Flac^{3D} bounded sub-domain, as shown in Figure 5.11, or in BEM infinite sub-domain, as shown in Figure 5.12. The vertical stress σ_z is plotted in both Flac^{3D} and BEM sub-domains to see the continuity of its values across the interface and to compare the coupled solution with the uncoupled Flac^{3D} solution with models of ratios R/a =12, R/a =3 and R/a =2 see Figures (5.13 up to 5.16); See also Figures 5.17 and 5.18 in Appendix b, p.293 and p.294, respectively. The solution's accuracy obtained by the coupled method surpassed its counterpart in the uncoupled Flac^{3D} solution for the vertical stress as shown in Tables 5.5 and 5.6 (see Appendix b, p.295 and p.296, respectively). The vertical stress σ_z on the boundary (the interface) at point B, see Figure 5.9, is obtained also, as a post processing outcome and as an advantage of the BEM over the FDM and FEM (see Table 5.6 in Appendix b, p.296). The obtained stress value was with a good agreement with the analytical solution.

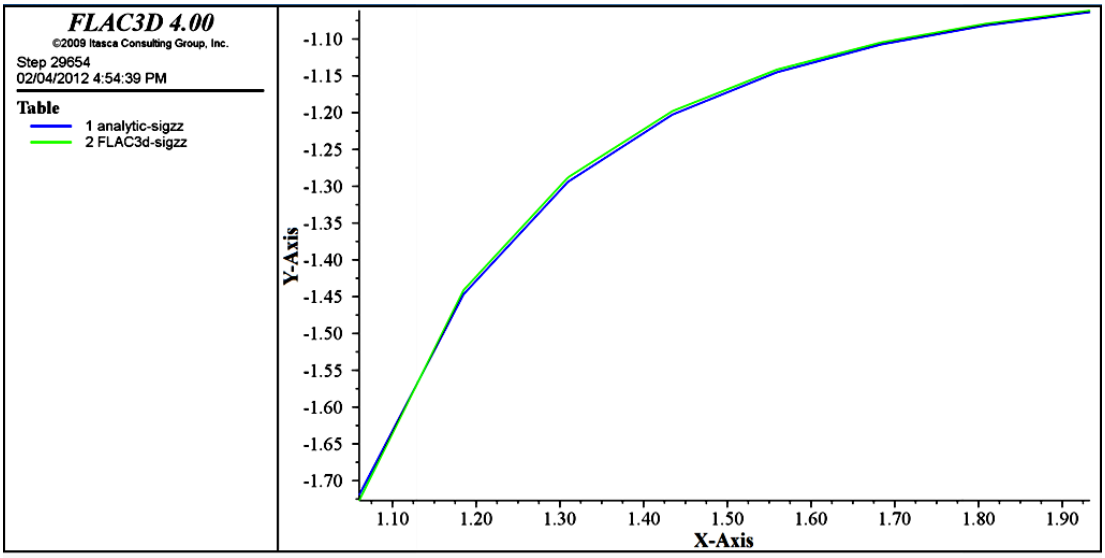


Figure 5.11 Coupled Flac3D-BEM and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in the Flac3D sub-domain, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

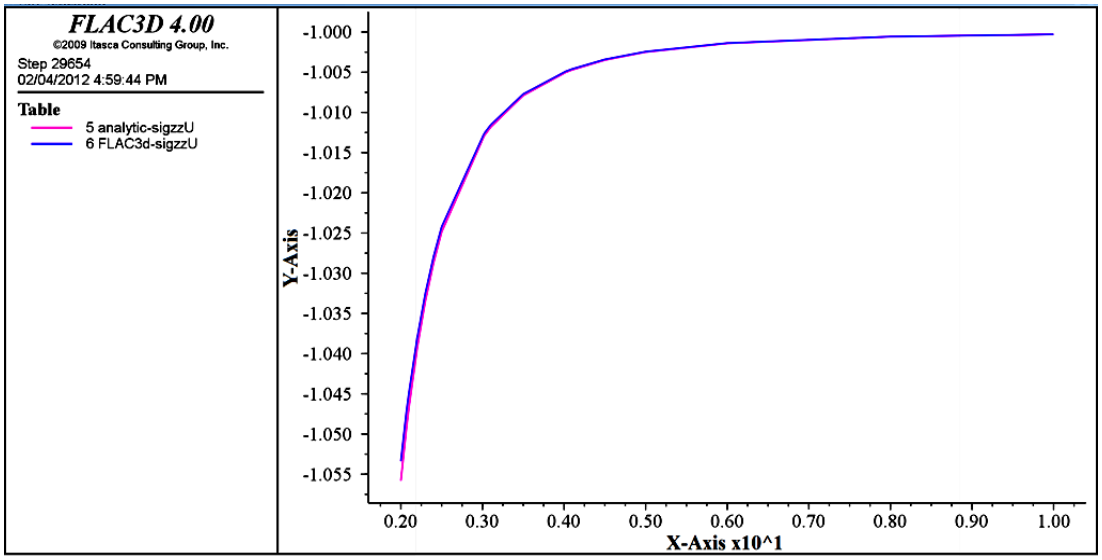


Figure 5.12 Coupled Flac3D-BEM and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in the BEM infinite sub-domain, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

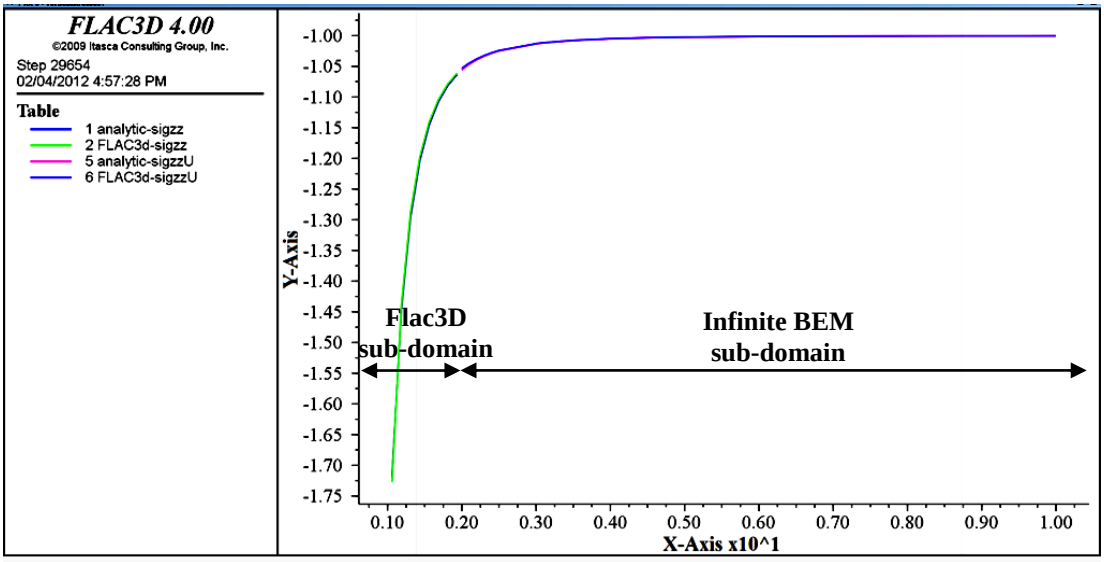


Figure 5.13 Coupled Flac3D-BEM and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in both Flac3D and BEM sub-domains, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

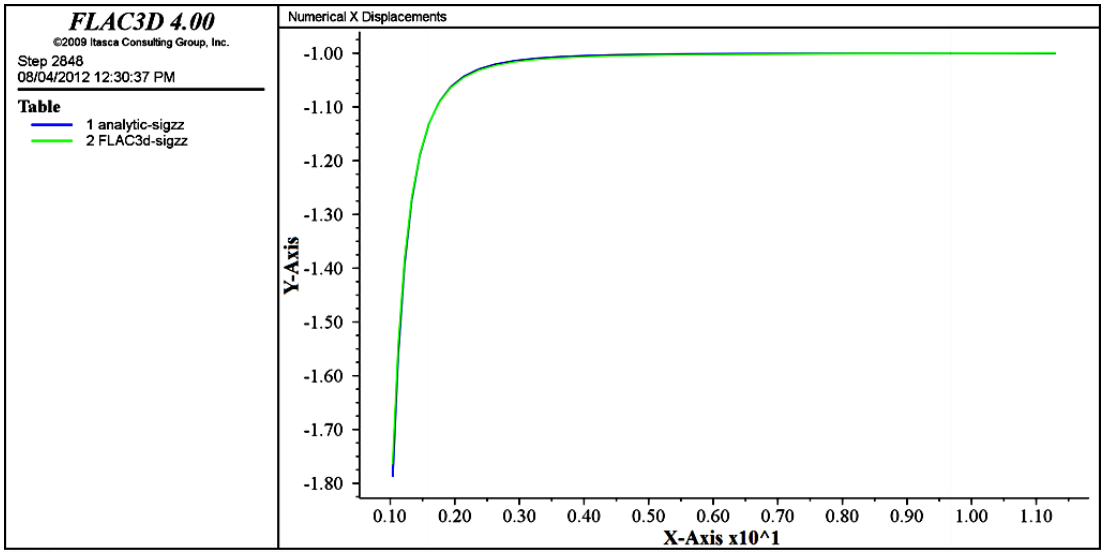


Figure 5.14 Uncoupled Flac3D and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in the Flac3D model of the ratio R/a =12, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

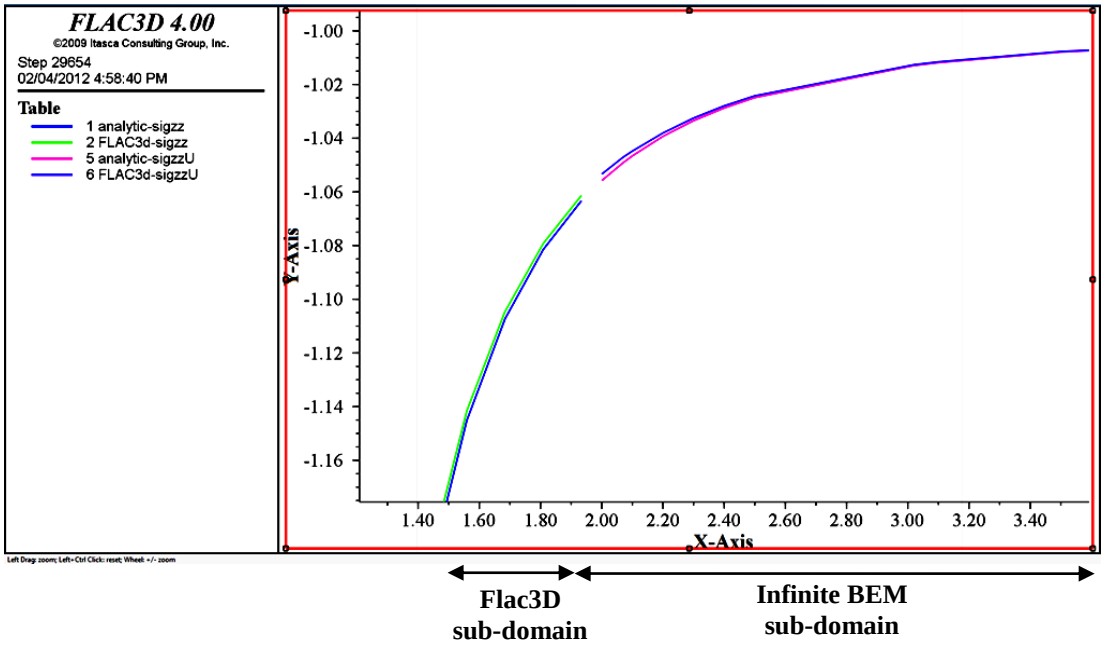


Figure 5.15 The Continuity of the coupled Flac3D-BEM vertical stress (σ_z) in plane (z = 0) across the Flac3D and BEM sub-domains' interface, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

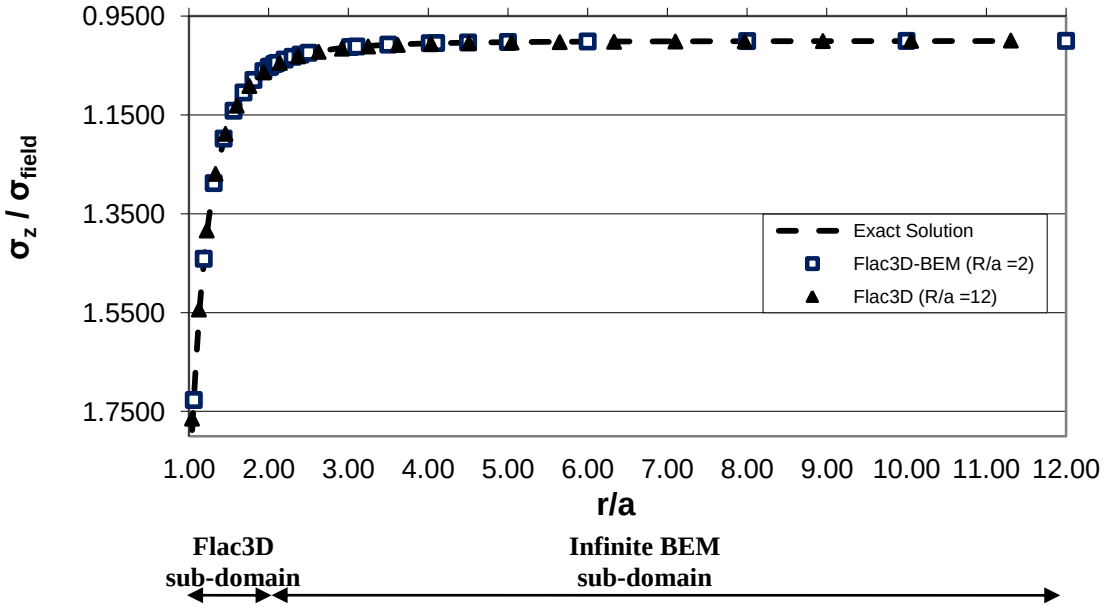


Figure 5.16 Coupled Flac^{3D} -BEM, uncoupled Flac^{3D} and exact vertical stress (σ_z) in plane (z = 0) as a function of (r/a) in both Flac^{3D} and BEM sub-domains

The horizontal displacement values in plane (z=0) computed by the coupled method converges very well asymptotically in comparison with the uncoupled Flac^{3D} solutions using models of ratios: R/a = 2 up to R/a =12 (see Figure 5.19). The displacement compatibility across the interface is further clear evidence of the accuracy of the coupled method (see Figure 5.20), which shows the horizontal displacement in both Flac^{3D} and BEM sub-domains and the computed displacement very close to the interface (r/R = 1.01) (see Table 5.7 in Appendix b, p.297).

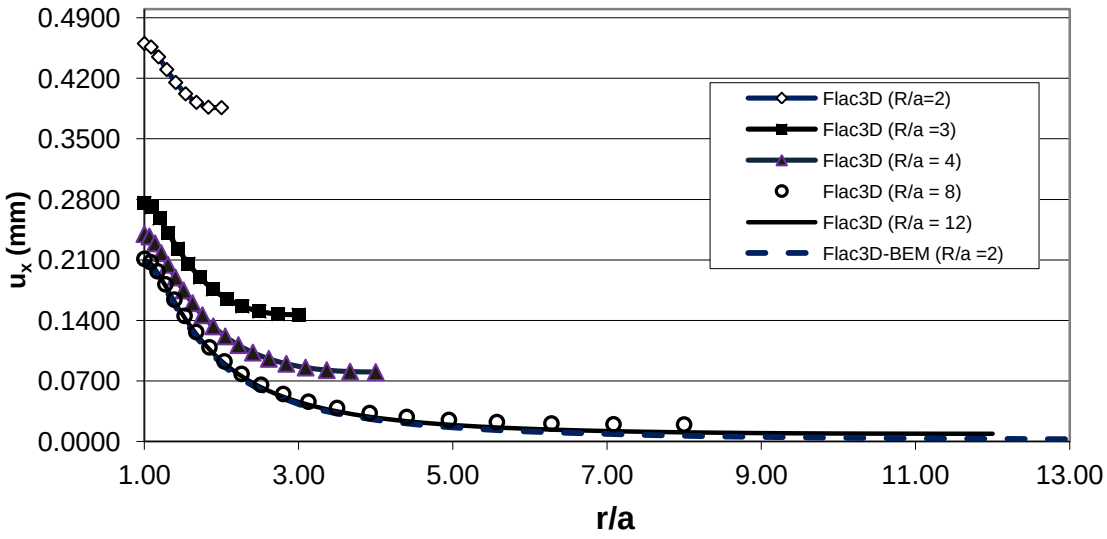


Figure 5.19 Coupled Flac^{3D} -BEM and uncoupled Flac^{3D} horizontal displacement u_x in plane (z = 0) as a function of (r/a) in both Flac^{3D} and BEM sub-domains

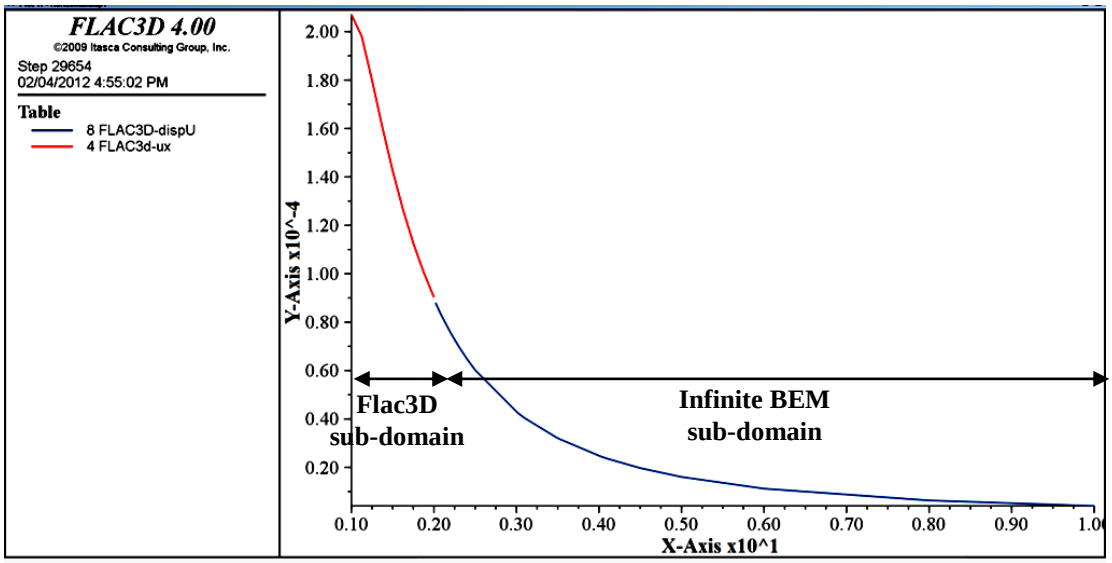


Figure 5.20 The Compatibility of the coupled Flac3D-BEM horizontal displacement u_x in plane (z = 0) across the Flac3D and BEM sub-domains' interface, (X-Axis ≡ r) and (Y-Axis ≡ u_x).

Convergence tests

1. A mesh size solution convergence test of the proposed coupling method is conducted for:
    1. The vertical displacement u_z at point A, as shown in Figure 2.22.
    2. The vertical stress σ_z at points H, G, E and C in the Flac^{3D} sub-domain and the horizontal displacement u_x at points I, G, F and D in the same sub-domain, as shown in Figure 2.23. Test points positions in plane (z = 0) are shown in Figure 2.21 and Table 5.8.
    3. The vertical stress σ_z at points B, K, L and M in the BEM sub-domain and the horizontal displacement u_x at points M, L and K are also positioned in the BEM sub-domain, as shown in Figure 2.24.

These tests proved that the solution is independent of the number of zones or the mesh size. This indicates that the suggested coupling method is efficient and reliable.

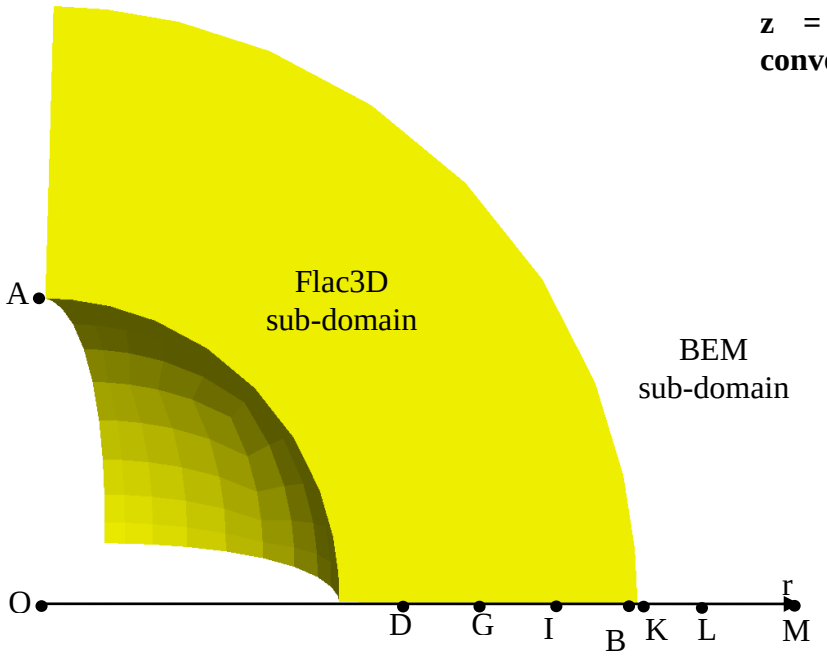


Figure 5. 21 Point positions for mesh size convergence test

Table 5.8 Point positions (plane z = 0) for the mesh size convergence test

	r	Point
Flac^{3D} sub-domain	1.2	C
	1.25	D
	1.3	E
	1.4	F
	1.5	G
	1.72	H
	1.75	I
BEM sub-domain	2	B
	2.02	K
	2.5	L
	3.5	M

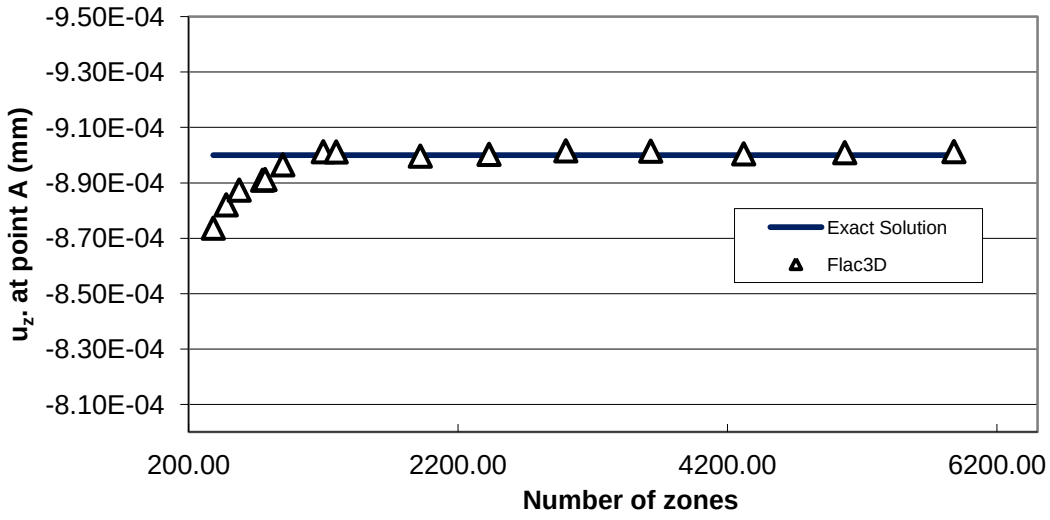
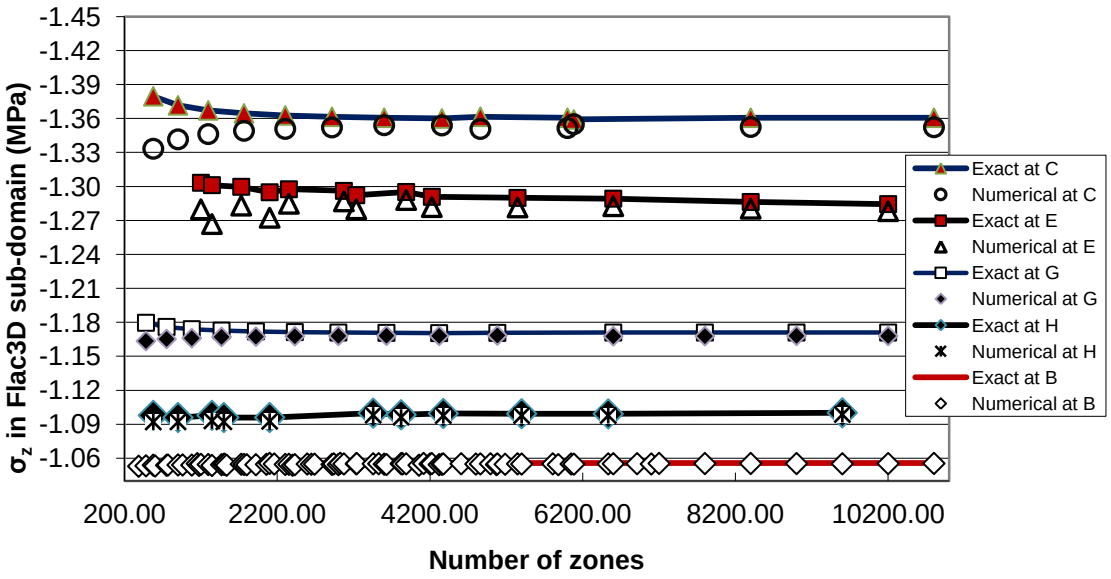


Figure 5.22 Mesh size convergence test for the vertical displacement u_z at point A

a.



b.

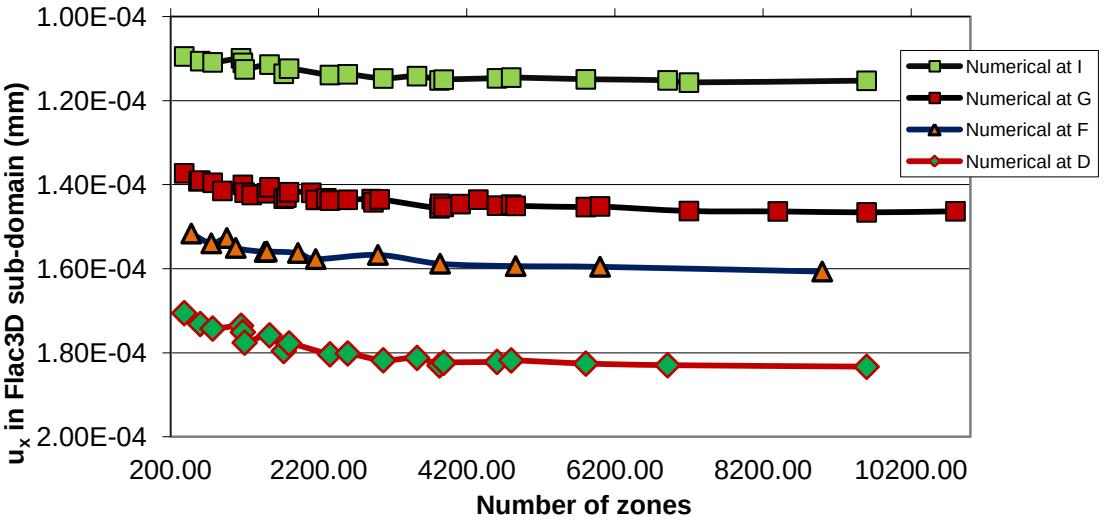
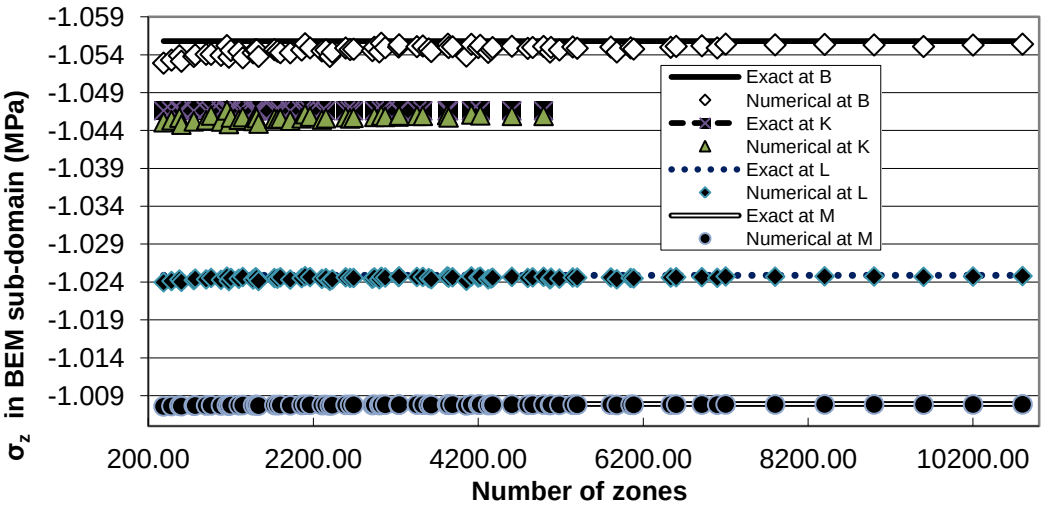


Figure 5.23 Mesh size convergence test in Flac^{3D} sub-domain for: a. Vertical stress σ_z and b. Horizontal displacement u_x

a.



b.

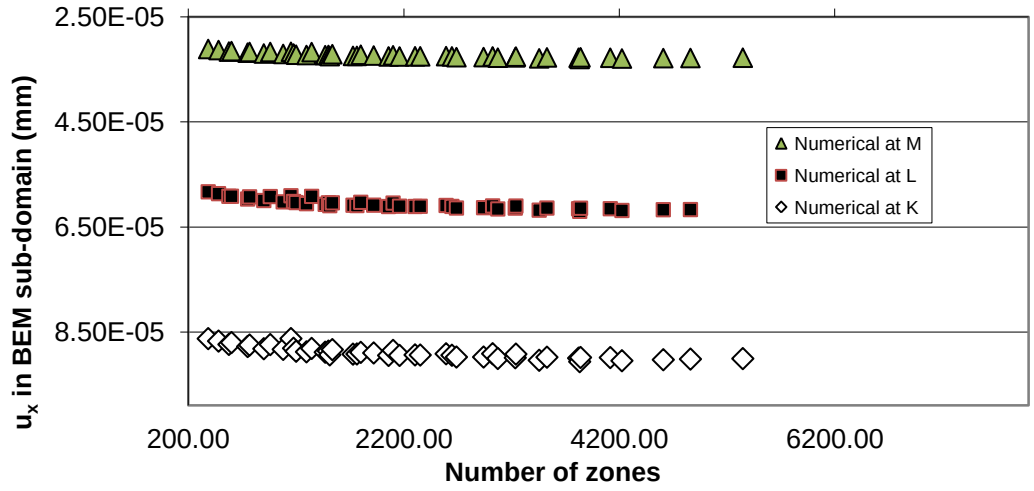


Figure 5.24 Mesh size convergence test in BEM sub-domain for:
a. Vertical stress σ_z and b. Horizontal displacement u_x

2. Iterations convergence test: the test is conducted to prove that the solution of the proposed coupling method is independent of the number of the renewal iterations. See in Table 5.9 and Figure 5.25 the test points and the zones' positions. The mesh size of the used Flac^{3D} model, shown in Figure 5.25, is 1200 zones.

Table 5.9 Points' positions (in plane z = 0) used for the iteration convergence test

	r	Zone Number
Flac^{3D} sub-domain	1.06	12
	1.19	14
	1.31	16
	1.43	18
	1.56	20
	1.68	22
	1.81	24
	1.93	26
BEM sub-domain	2.00	104
	2.02	53
	2.07	56
	2.10	59
	2.20	62
	2.30	65
	2.40	68
	2.50	71
	3.02	74
	3.10	77
	3.50	80
	4.02	83
	4.10	86
	4.50	89
	5.00	92
	6.00	95
	8.00	98
	10.00	101

	r	Point Number
Flac^{3D} sub-domain	1.00	44
	1.13	45
	1.25	46
	1.38	47
	1.50	48
	1.63	49
	1.75	50
	1.88	51
BEM sub-domain	2.00	52
	2.02	55
	2.07	58
	2.10	61
	2.20	64
	2.30	67
	2.40	70
	2.50	73
	3.02	76
	3.10	79
	3.50	82
	4.02	85
	4.10	88
	4.50	91
	5.00	94
	6.00	97
	8.00	100
	10.00	103

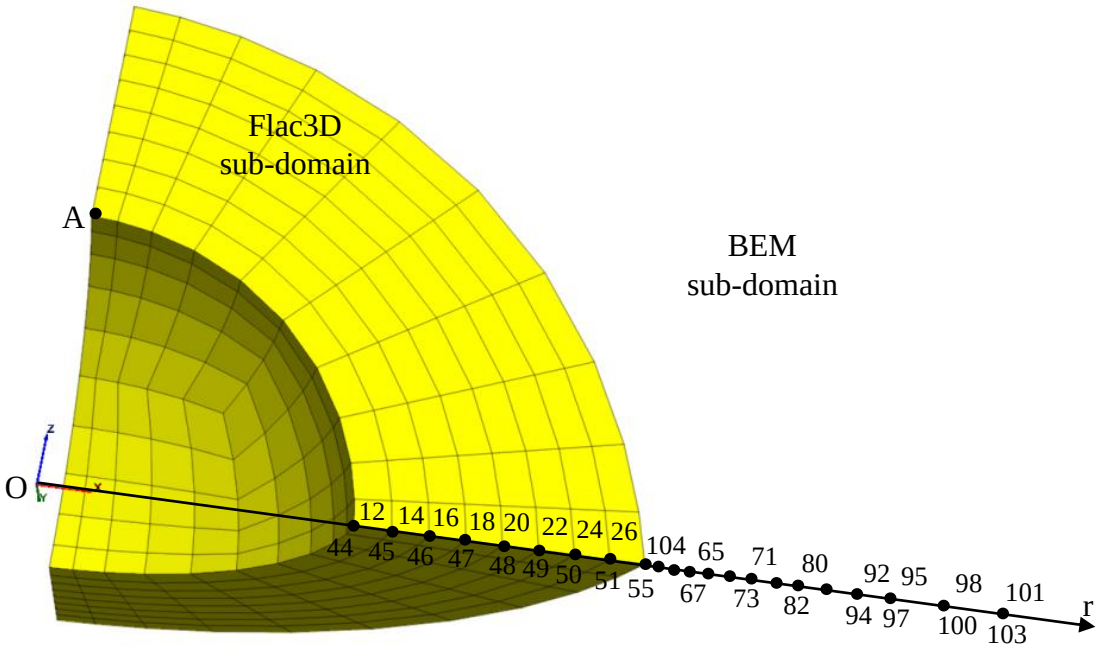
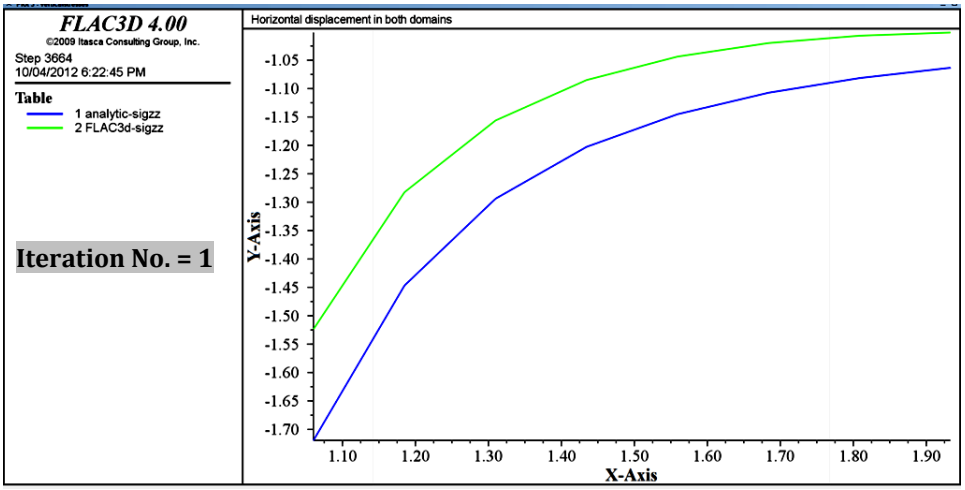


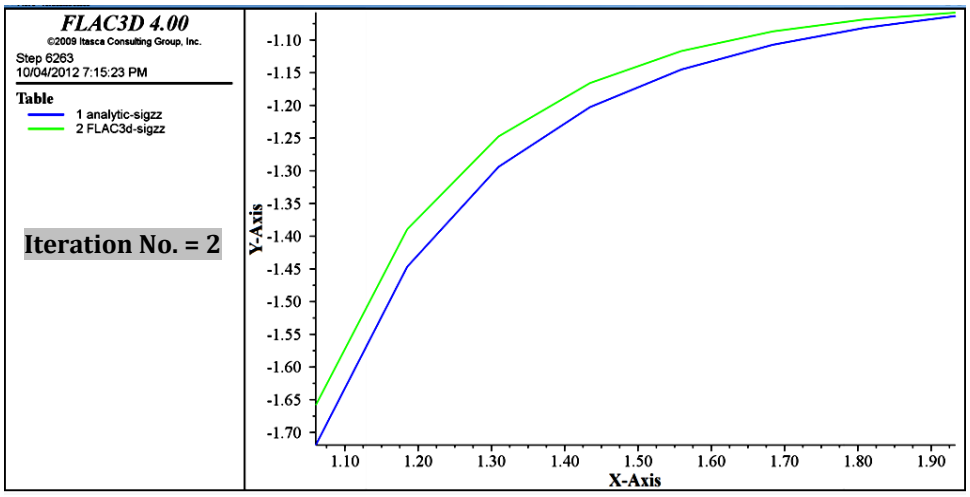
Figure 5. 25 Zones and points' positions used for the number of iterations convergence test

The coupled Flac^{3D} -BEM stress (σ_z) in plane (z = 0) in the Flac^{3D} and BEM sub-domains tends to converge fast starting from iteration 3 or 4, as shown in Figures 5.26 and 5.27. The stress continuity across the interface starts to form at the same third or fourth iteration as seen in Figure 5.28, whereas the horizontal displacement u_x compatibility starts to form later in the scheme at the sixth or seventh iterations as shown in Figure 5.29.

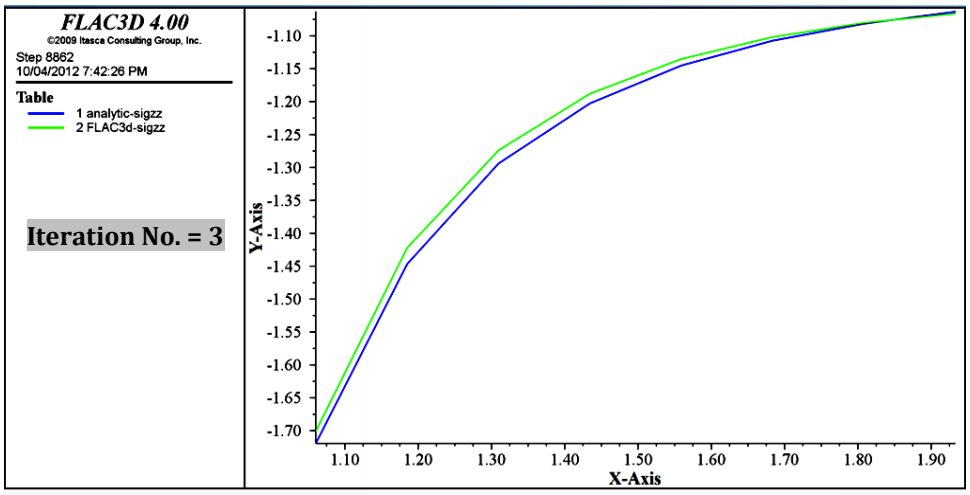
a.



b.



c.



d.

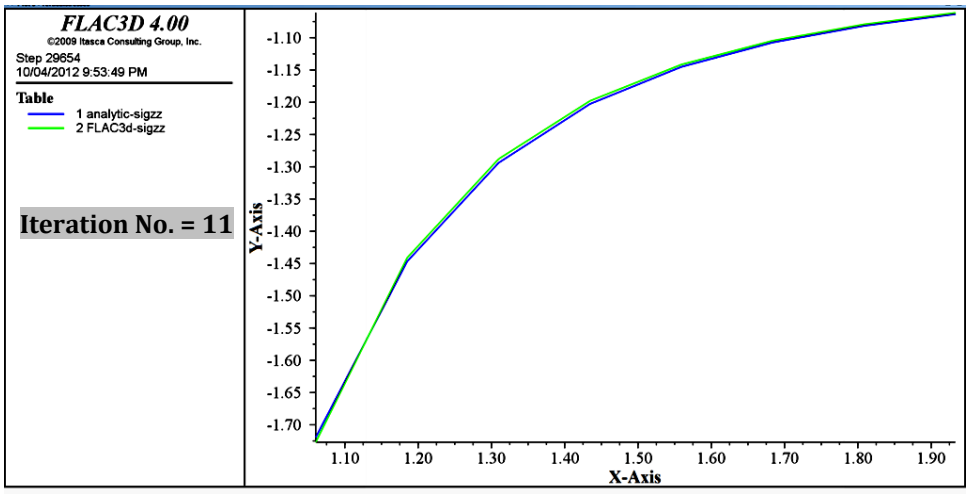
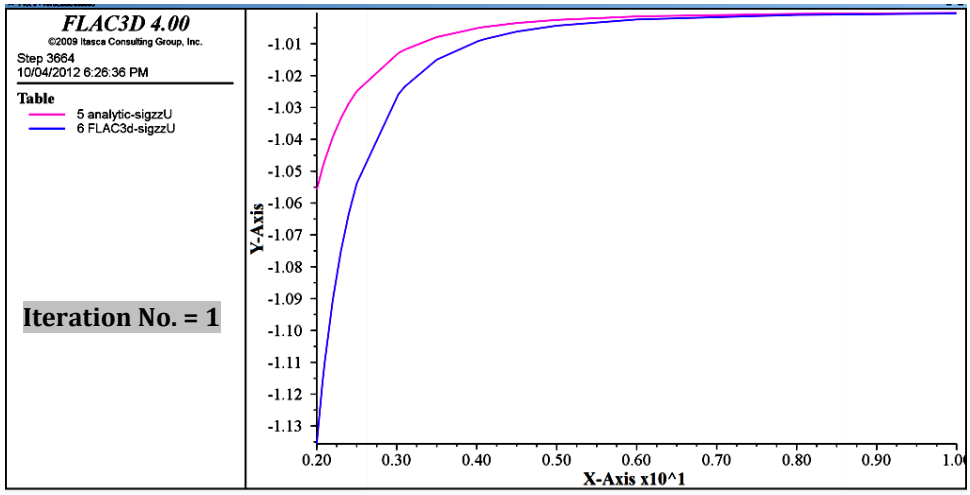
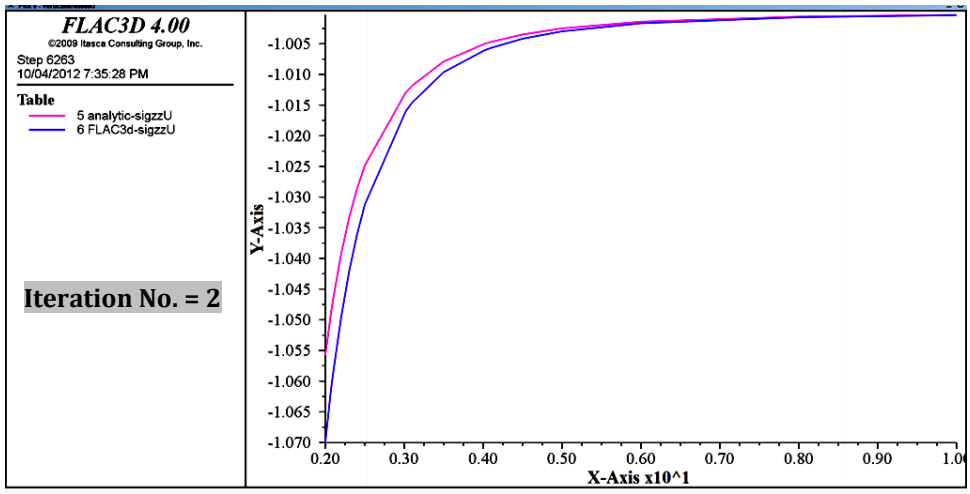


Figure 5.26 Development of coupled Flac3D-BEM stress (σ_z) in the Flac3D sub-domain (in plan z=0) over the iterative scheme, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

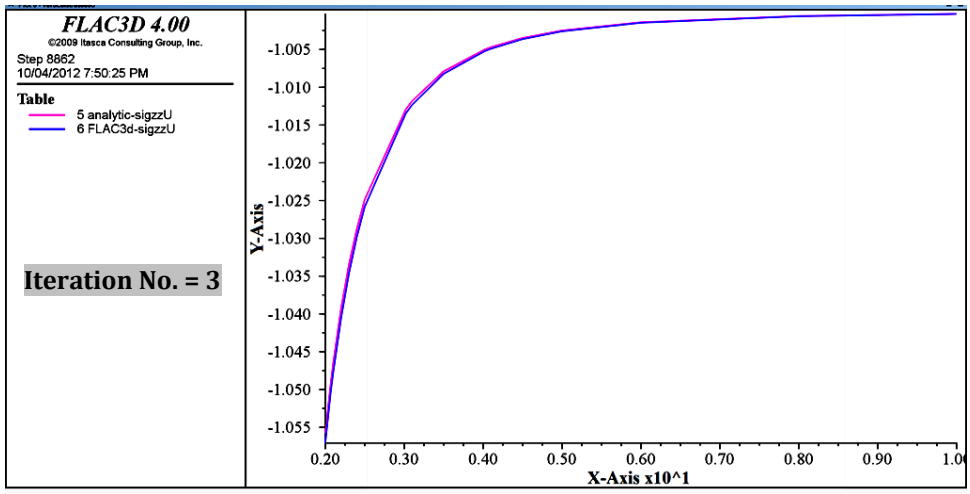
a.


FLAC3D 4.00
©2009 Itasca Consulting Group, Inc.
Step 3664
10/04/2012 6:26:36 PM
Table
5 analytic-sigzzU
6 FLAC3d-sigzzU
Iteration No. = 1
Y-Axis
-1.01
-1.02
-1.03
-1.04
-1.05
-1.06
-1.07
-1.08
-1.09
-1.10
-1.11
-1.12
-1.13
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
X-Axis x10^1

b.


FLAC3D 4.00
©2009 Itasca Consulting Group, Inc.
Step 6263
10/04/2012 7:35:28 PM
Table
5 analytic-sigzzU
6 FLAC3d-sigzzU
Iteration No. = 2
Y-Axis
-1.005
-1.010
-1.015
-1.020
-1.025
-1.030
-1.035
-1.040
-1.045
-1.050
-1.055
-1.060
-1.065
-1.070
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
X-Axis x10^1

c.


FLAC3D 4.00
©2009 Itasca Consulting Group, Inc.
Step 8862
10/04/2012 7:50:25 PM
Table
5 analytic-sigzzU
6 FLAC3d-sigzzU
Iteration No. = 3
Y-Axis
-1.005
-1.010
-1.015
-1.020
-1.025
-1.030
-1.035
-1.040
-1.045
-1.050
-1.055
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
X-Axis x10^1

d.

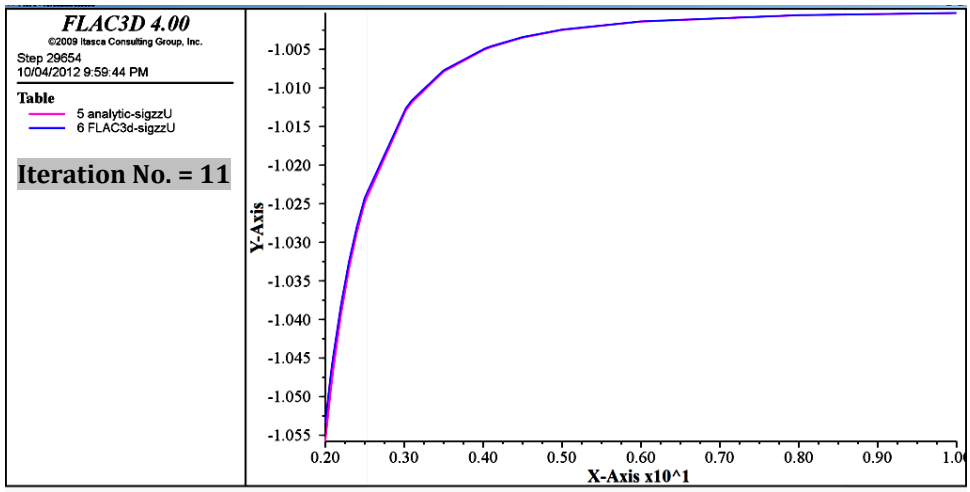
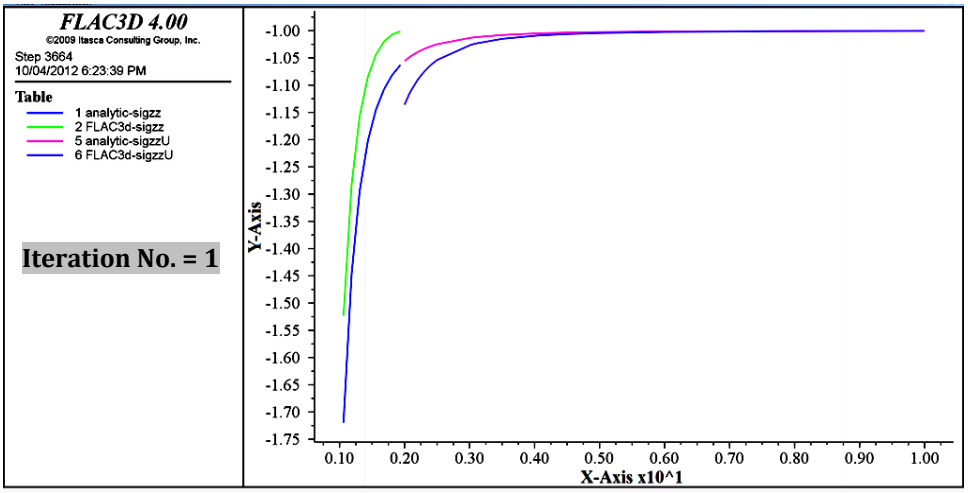
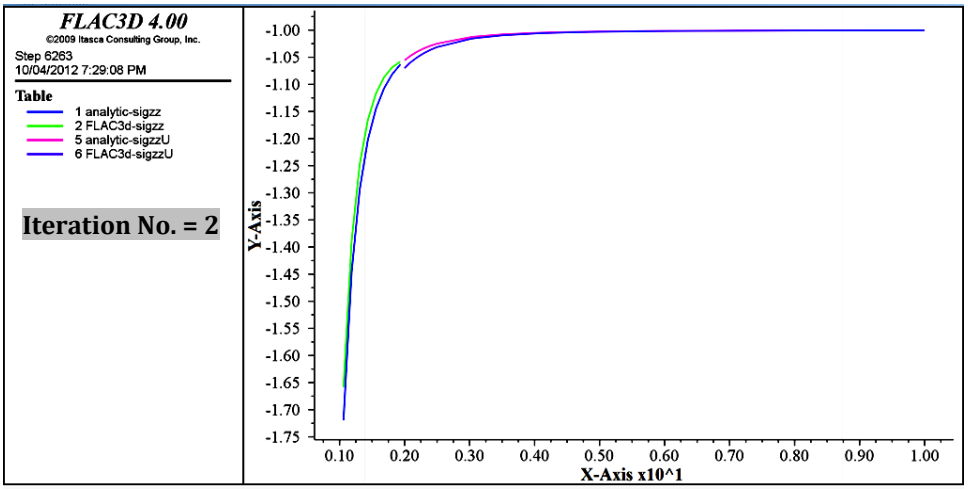


Figure 5.27 Development of coupled Flac3D-BEM stress (σ_z) in the BEM sub-domain (in plan z=0) over the iterative scheme, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

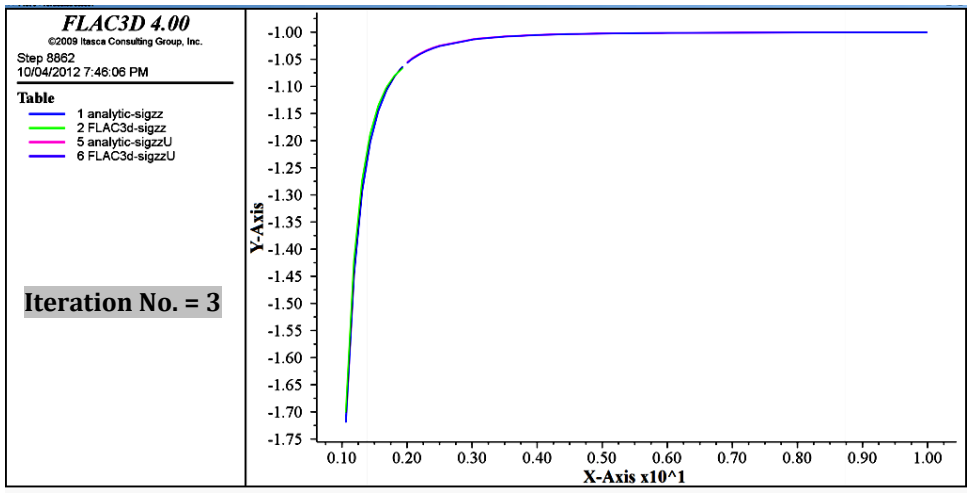
a.



b.



c.



d.

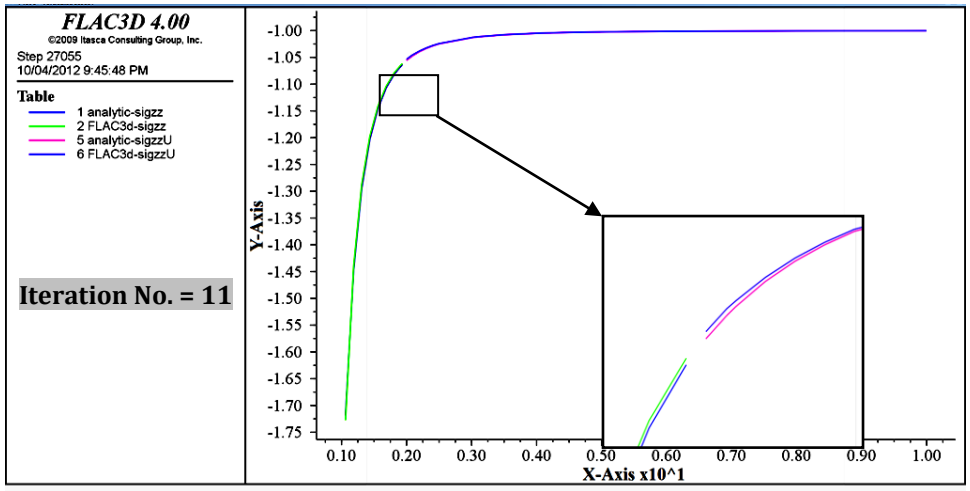
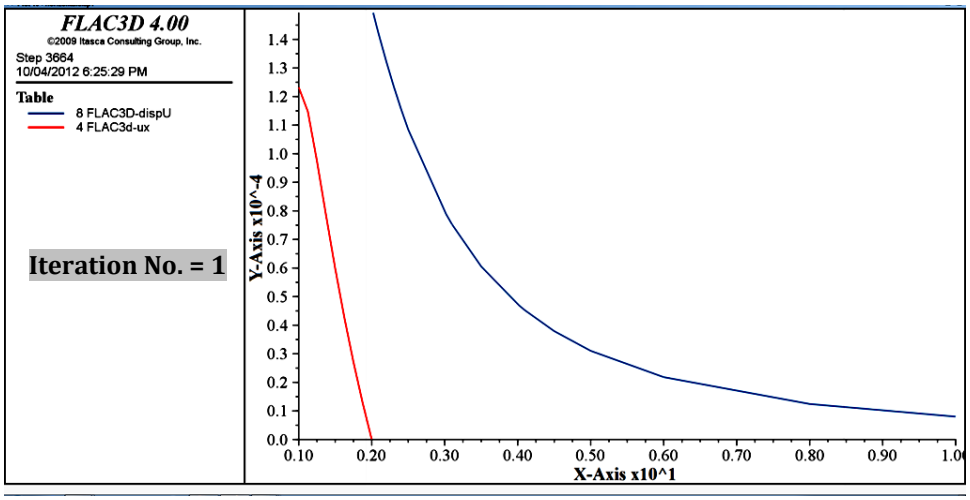
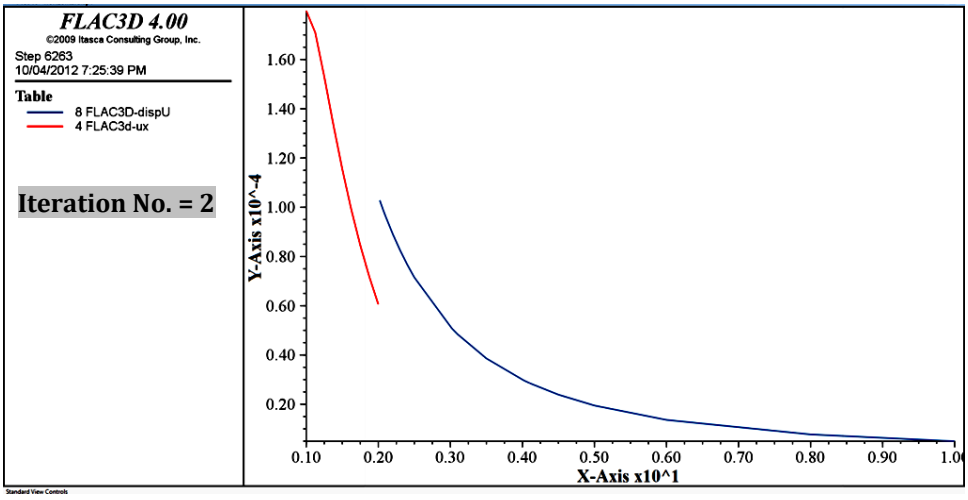


Figure 5.28 Development of the stress (σ_z) continuity across the interface over the iterative scheme, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z)*.

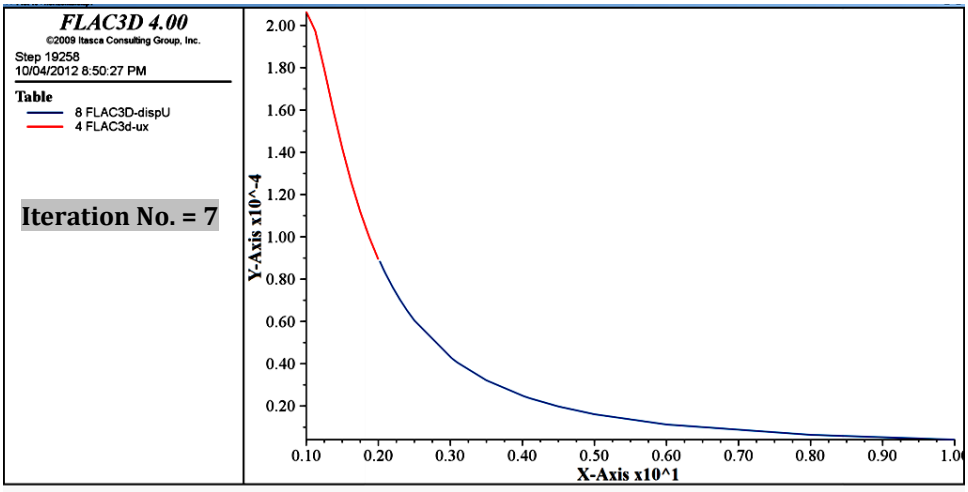
a.



b.



c.



d.

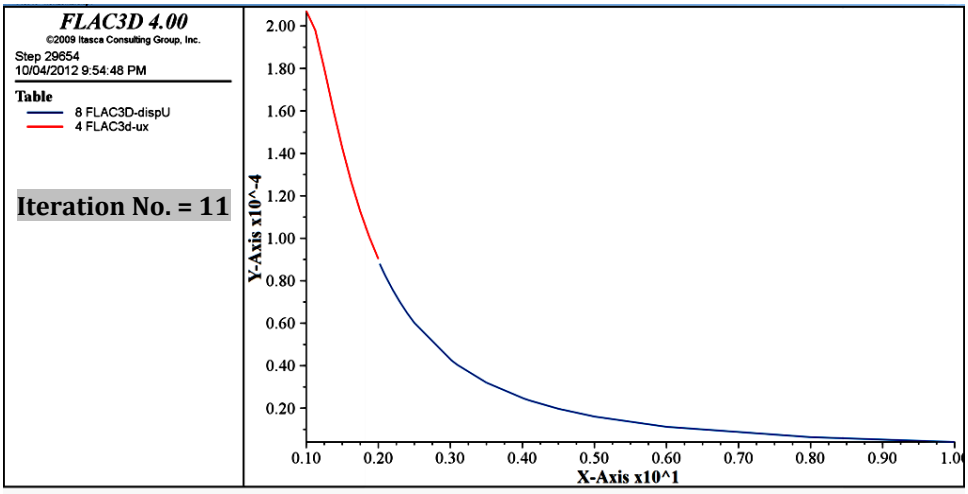
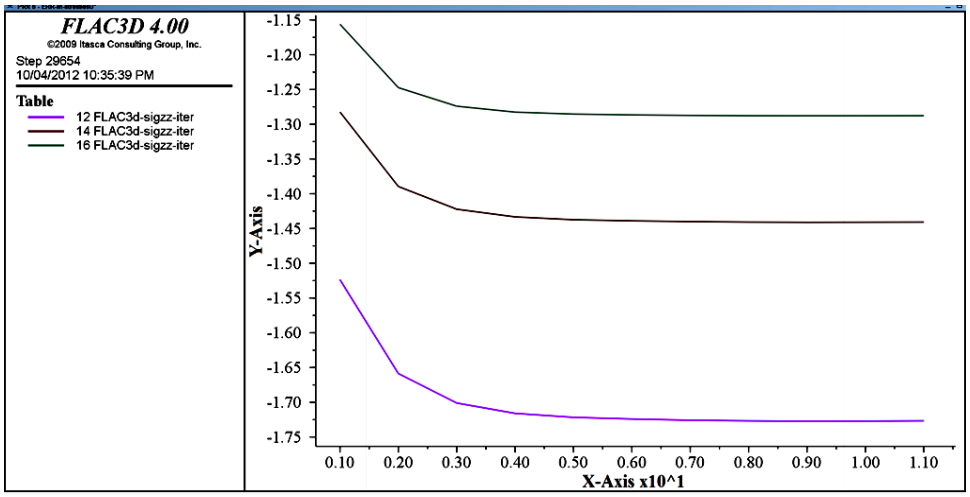


Figure 5.29 Development of the displacement (u_x) compatibility across the interface over the iterative scheme, *(X-Axis ≡ r) and (Y-Axis ≡ u_x)*.

The stable values of the vertical displacement u_z at point A as shown in Figure 5.32 (see Appendix b, p. 299), the vertical stress at the number of zones as visualized in Figures 5.30 and 5.31 (see the latest in Appendix b, pp. 298-299), and the horizontal displacement at the number of points in both Flac^{3D} and BEM sub-domains as seen in Figures 5.33 and 5.34 indicate that the suggested coupling method converges successfully with a rapid rate (see Appendix b in p.300 and pp. 300-301, respectively). However, the stress tends to converge faster than the displacement as observed in the above named figures.

a.



b.

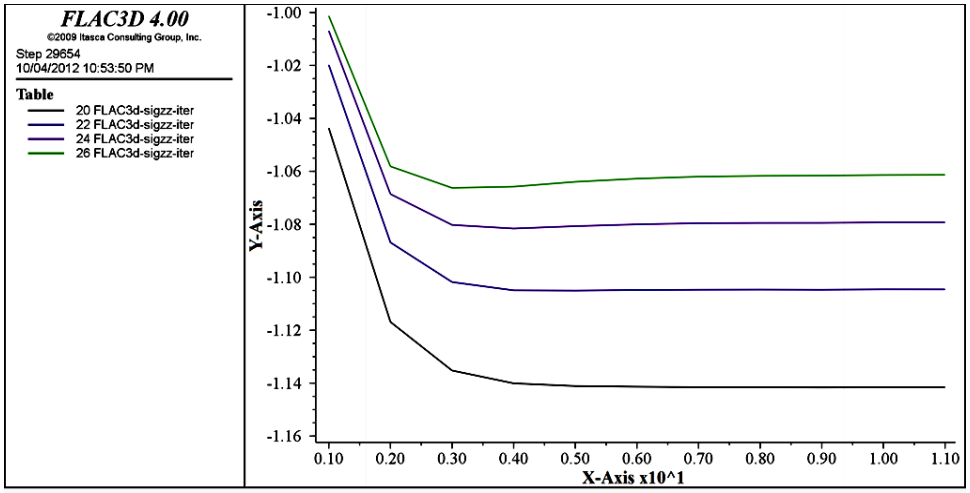


Figure 5.30 Convergence of stress (σ_z) at chosen zones in the Flac3D sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ σ_Z).*

If the same problem is solved, using the Flac^{3D} model defined earlier in this section with three different mesh sizes: mesh size b as shown in Figure 5.25, mesh size a and c listed in Table 5.10, the minimum number of iterations [80] (NIT) needed to reduce the initial error by a factor $\eta = 10^{-2}$:

$$\left\|\mathbf{e}_{,\mathrm{NIT}}\right\| < \eta \left\|\mathbf{e}_{,0}\right\| \tag{5.2}$$

Where: $e_{,n}$ is the error at iteration n of the nodal displacement at the interface,

is observed to be independent of the number of zones (number of degrees of freedom) and the average reduction factor per iteration (E. R. F.) [80]:

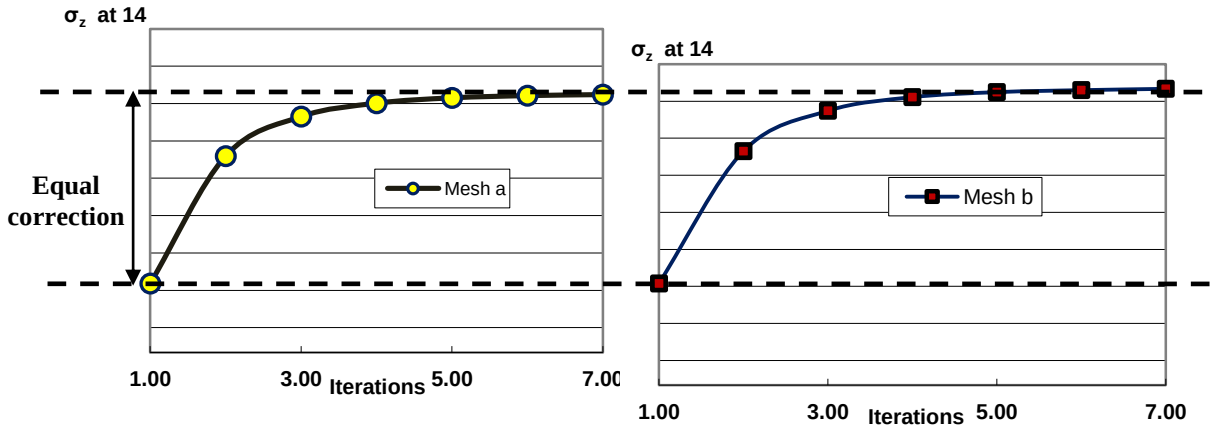
$$\mathrm{ERF} = \left(\left\|\mathbf{e}_{,n}\right\| / \left\|\mathbf{e}_{,0}\right\|\right)^{1/n} \tag{5.3}$$

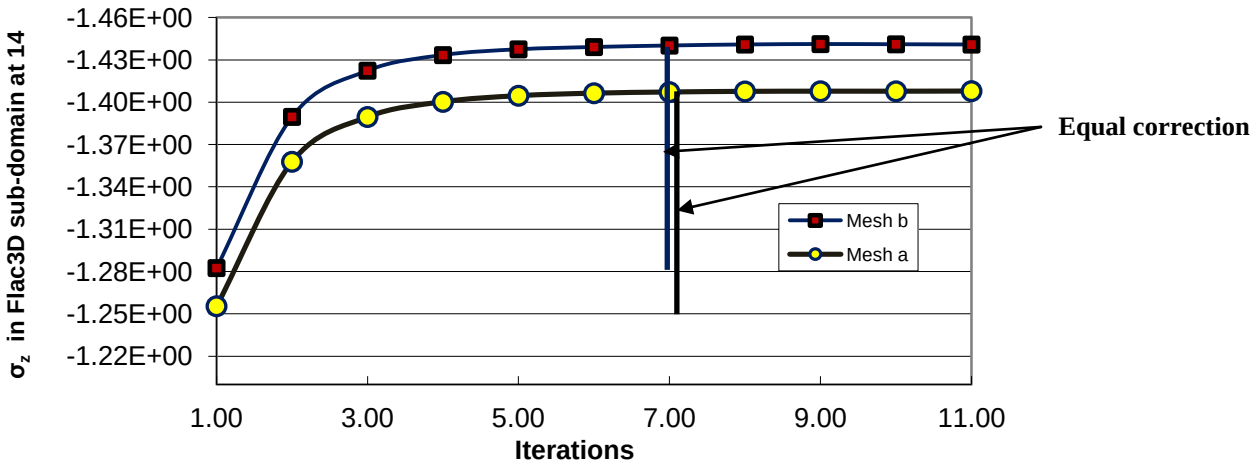
is almost similar in the three cases (see Table 5.10 and Figure 5.35). See also Figures 5.36 up to 5.40 in Appendix b, pp. 302-304.

Table 5.10 Average reduction factors for different mesh sizes

Mesh size (No. Of zones)	Mesh name	N.I.T.	E. R. F.
768	a	7	0.4758
1200	b	7	0.4777
3072	c	7	0.4865

I.

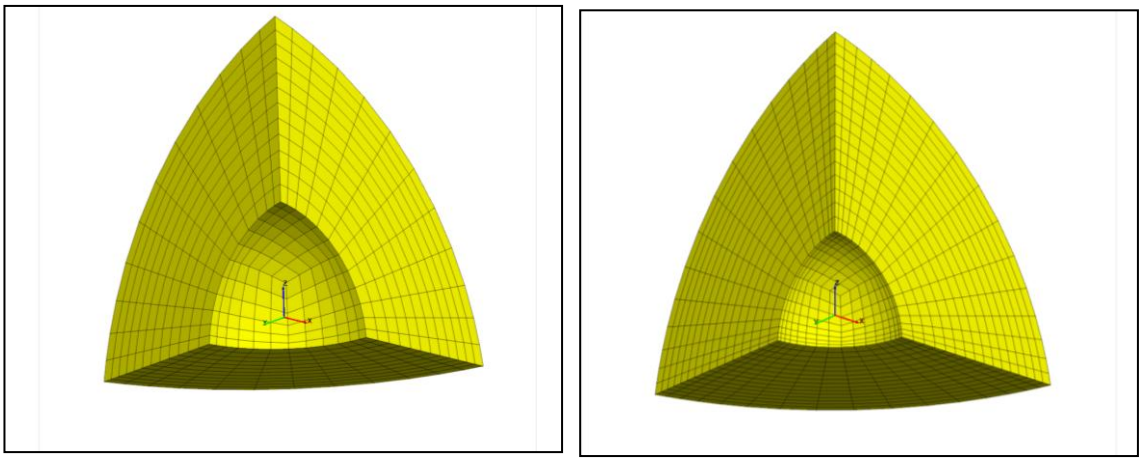




II.

Figure5. 35 I. and II. Evolution of stress (σ_z) at zone No. 14 in the Flac^{3D} sub-domain over the iterative scheme for meshes a and b.

Truncation boundary position effect: to check if this effect is eliminated, the problem is solved again using a Flac^{3D} model of different ratios (see Figure 5.41), using the proposed coupling method. The solution obtained with models of ratioes 2, 3 and 4 is independent from the position of truncation boundary, as shown in Figures 5.42 up to 5.47 (see Figures 5.42 and 5.46 in Appendix b in p.306 and p.307, respectively). The vertical stress σ_z continuity and the horizontal displacement u_x compatibility across the interface are also achieved as Figures 5.42 and 5.46 depict. A less than 1% RE with the exact solution for the vertical stress σ_z in plane (z = 0) and the vertical displacement u_z at point A is observed in Figures 5.42 up to 5.45 and in Tables 5.11 up to 5.13 (see Tables 5.11 and 5.12 in Appendix b in p.304 and p.305, respectively).



a. **b.**

Figure 5.41 Flac3D Models of ratios: a. R/a = 3 b. R/a = 4

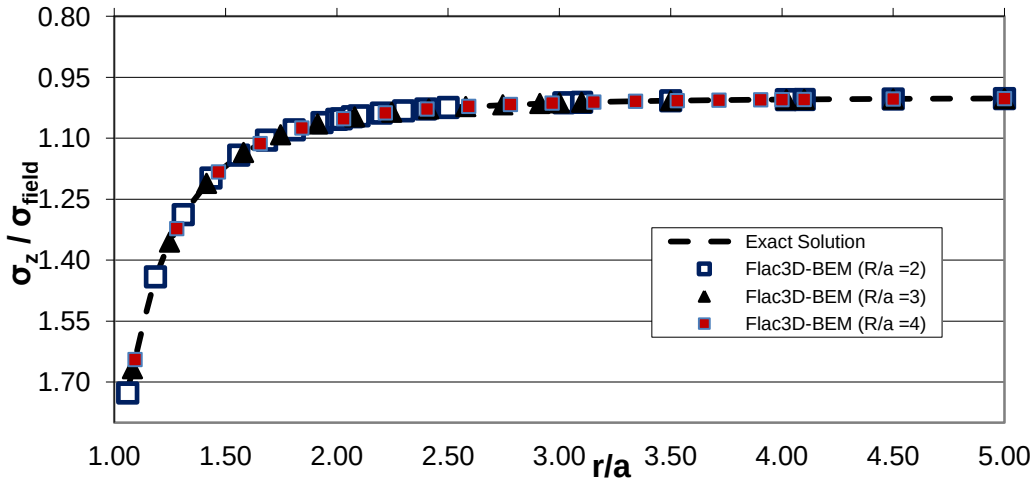


Figure 5.43 Coupled and exact vertical stress (σ_z) in plane (z = 0) both Flac3D and BEM sub-domains.

Table 5.13 Coupled Flac^{3D} -BEM and exact vertical displacement values at point A for different Flac^{3D} model size

R/a	Number of Zones	CPU (seconds)	u_z Numerical (mm)	u_z Exact (mm)	R. E. %
2	1200	38.8	-0.9065	-0.9	0.68
3	2592	81.8	-0.9045	-0.9	0.50
4	6144	246.1	-0.9038	-0.9	0.43

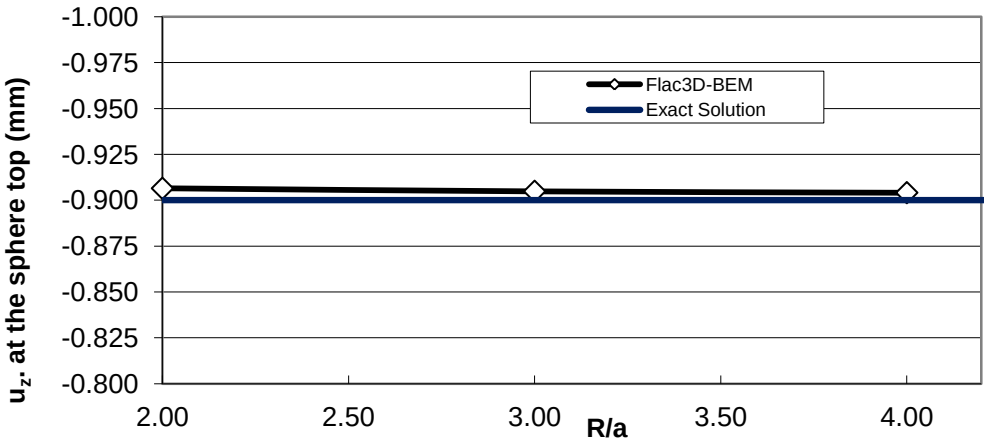
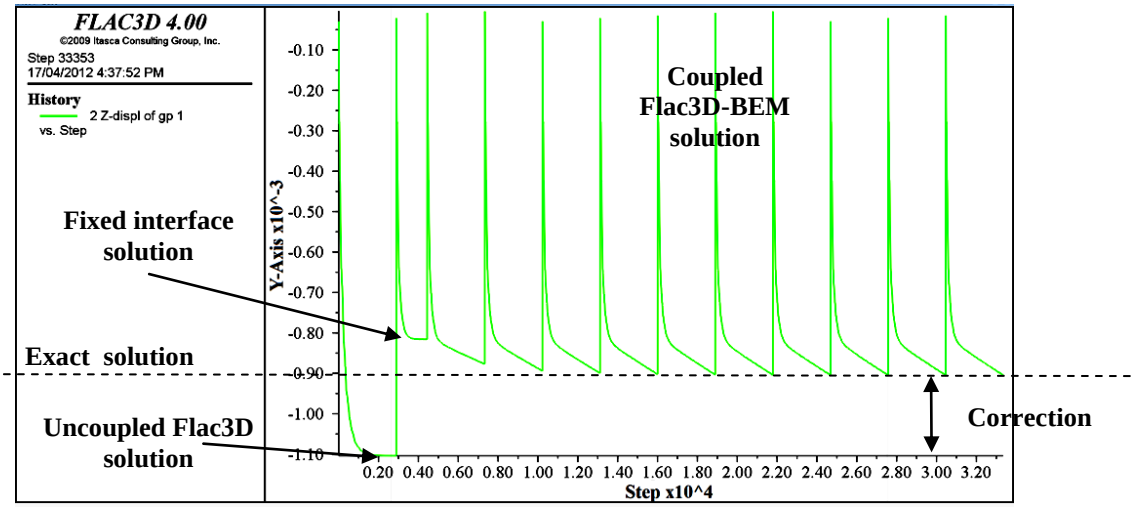
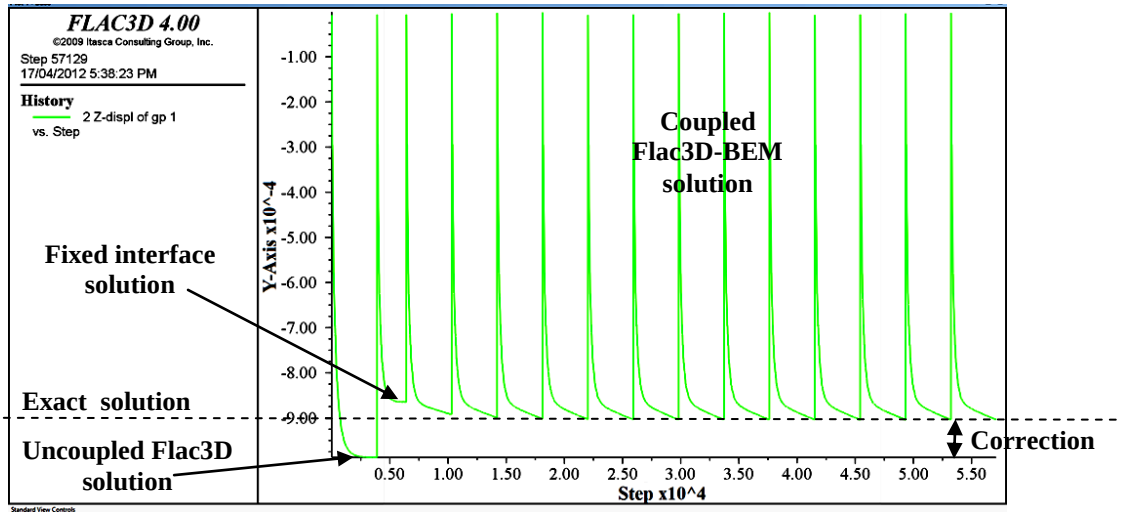


Figure 5.44 Constant value of coupled vertical displacement u_z at point A



a.



b.

Figure 5.45 History of Flac3D and coupled vertical displacement at point A, *(Y-Axis ≡ u_Z)*. The Flac3D model ratio: a. R/a = 3, b. R/a = 4

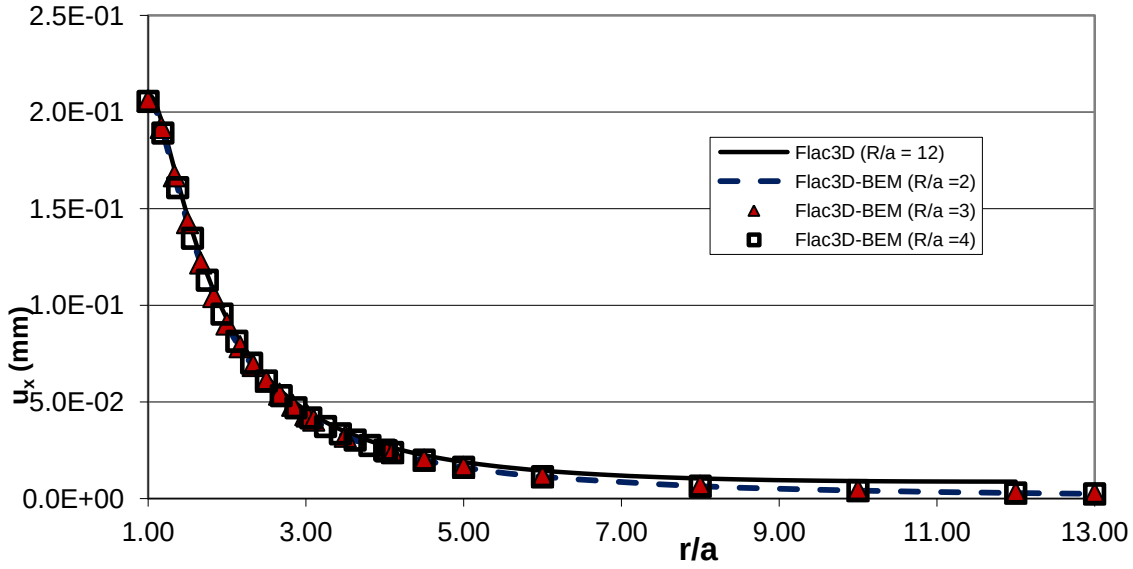


Figure 5.47 Coupled horizontal displacement u_x in plane (z = 0) in both Flac^{3D} and BEM sub-domains

It is important to observe that the runtime CPU increases significantly, as shown in Table 5.13, by increasing the model length (the mesh size) with no increase in the solution accuracy.

Material properties effect: To check also the solution independence from the material properties, Poison ratio υ and Young Modulus E, the problem is solved again using different material properties values (i.e. υ =0.3 and E = 2000 MPa). Figures 5.48 and 5.50 show an almost exact agreement (less than 1% RE) between the exact and the coupled solution for the vertical stress σ_z and vertical displacement u_z. The compatibility in horizontal displacement u_x in plane (z =0) across the interface is also fully developed as shown in Figure 5.49.

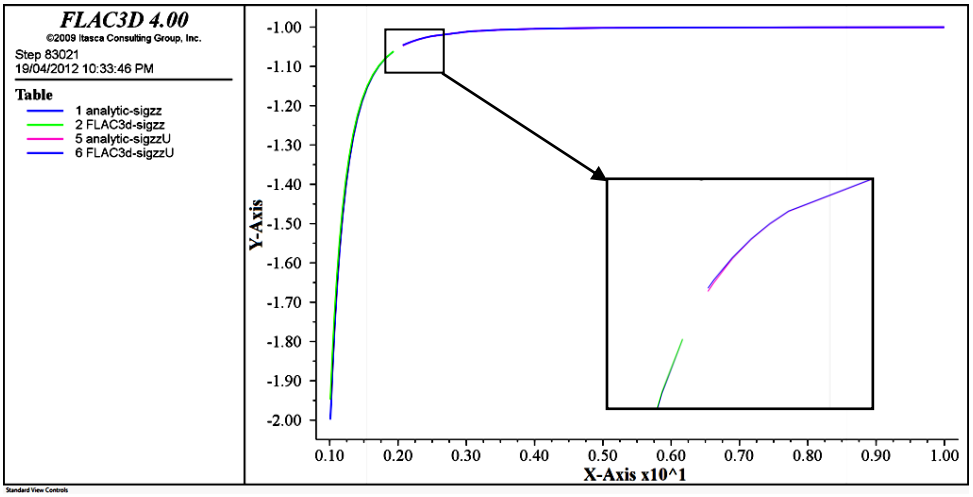


Figure 5.48 Coupled and exact vertical stress (σ_z) in plane (z = 0) in both Flac3D and BEM sub-domains, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).* R/a = 2

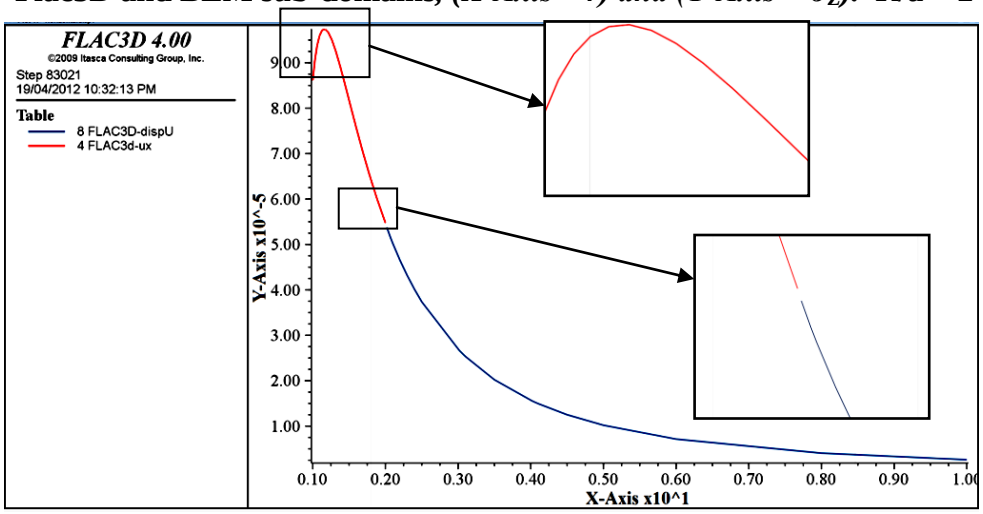


Figure 5.49 Compatibility of coupled horizontal displacement u_x in plane (z = 0) across the interface, (X-Axis ≡ r) and (Y-Axis ≡ u_x).

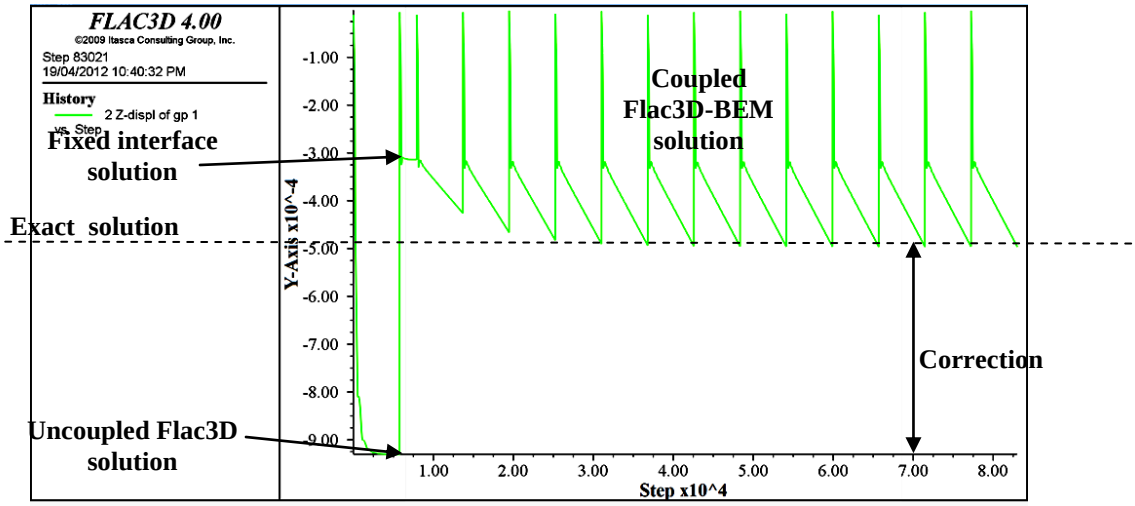


Figure 5.50 History of Flac3D and coupled vertical displacement at point A, *(Y-Axis ≡ u_Z).*

Effect of relaxation parameter and initial guess: The solution converges if the relaxation parameter is chosen in the range (0.1-0.6), as shown in Figure 5.51 and Table 5.14. The relaxation parameter optimum values which reduce the iteration number, needed to achieve the solution convergence, to the minimum, in the range of 0.45-0.6, provided the assumed initial guess ($\{\mathbf{u}^{I}_{F,0}\}$) equals zero (fixed interface) or close to zero.

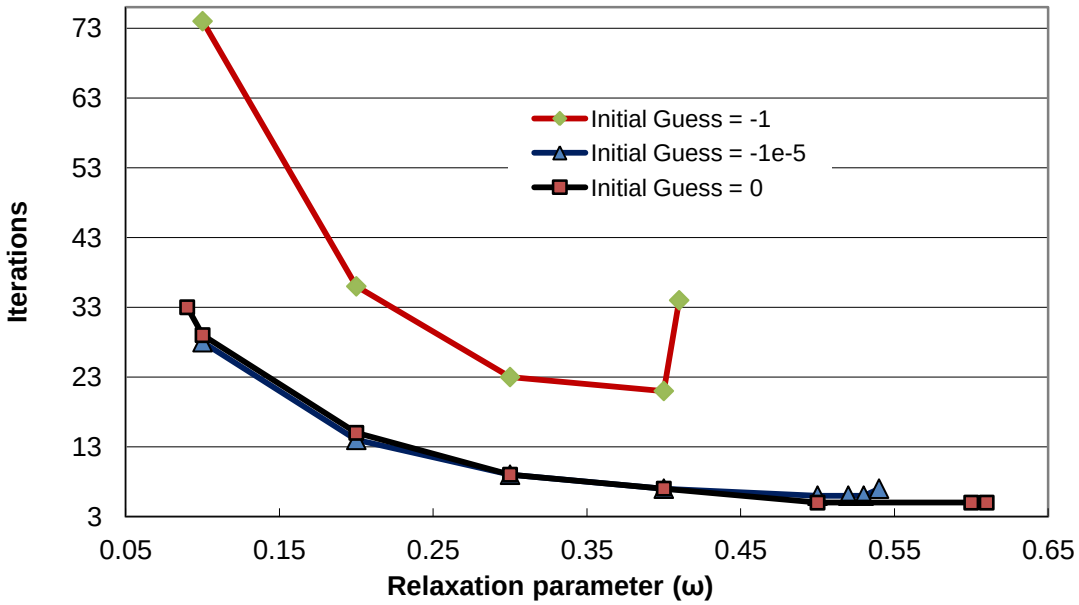


Figure 5.51 Effect of relaxation parameter and initial guess on the solution convergence

However, the initial guess has no effect on the solution convergence, but it increases the number of iterations if it has been given significantly large values. Moreover, the relaxation parameter optimum value range has reduced to (0.2-0.4) for initial guess as large as -1 (m), (see Figure 5.51). Because there are a large number of possible values for the relaxation parameter, the dynamic method proposed by Lin et al. [51] to obtain the optimal value for ω, is tested (see section 4.2.2, p. 110). Unfortunately, no converging solution is obtained by this method. The runtime (CPU) required to solve the problem with a Flac^{3D} model has the ratio R/a = 2 and 1200 zones on a 3.2 GHz i7 computer equals 28.72 seconds (see Figure 5.25). The minimum

number of iterations was 5 and the optimum value used for the relaxation parameter ω was 0.6. This time is less than the runtime required to obtain an uncoupled Flac^{3D} solution with a model (R/a = 12). The accuracy in the coupled solution exceeded the uncoupled one (see Table 5.4).

Table 5.14 Effect of relaxation parameter and initial guess on the solution convergence

Initial guess (m)	Relaxation parameter	No. of Iteration	ERF
0	0.61	5	0.3276
	0.6	5	0.3195
	0.5	5	0.3785
	0.4	7	0.4777
	0.3	9	0.5890
	0.2	15	0.7168
	0.1	29	0.8531
	0.09	33	0.8685
-1.00E-05	0.54	7	0.5130
	0.53	6	0.4555
	0.52	6	0.4396
	0.5	6	0.4106
	0.4	7	0.4763
	0.3	9	0.5896
	0.2	14	0.7110
	0.1	28	0.8496
-1	0.41	34	0.7723
	0.4	21	0.6585
	0.3	23	0.6759
	0.2	36	0.7796
	0.1	74	0.8875

5.3.1.4 Coupled Flac^{3D} -BEM Solution (Second Method/SDDM)

So far, the suggested coupling method (the first method/IDDM) has proven to obtain an accurate solution, but it relies on the user's experience and a trial and error scheme to obtain the required relaxation parameter optimum value to reduce the number of iterations and the CPU to the minimum. The author proposes a relaxation-parameter-independent- coupling-method which achieves the solution with a reduced runtime that competes with the optimum solution produced by the first method. The sequential Dirichlet-Neumann single step domain decomposition method's (SDDM) algorithm is:

For n =0, 1,

1. Set $\{\mathbf{u}^{I}_{F,0}\} = 0$,

2. Obtain $\{\mathbf{f}^{I}_{F,n}\}$ in the FD sub-domain using Flac^{3D} program according to the program algorithm detailed in section 3.5.

3. Obtain $\{\mathbf{t}^{I}_{B,n}\}$ according to equation $\{\mathbf{f}^{I}_{F,n}\} + [\mathbf{M}]\{\mathbf{t}^{I}_{B,n}\} = 0$,

4. Obtain $\{\mathbf{u}^{I}_{B,n}\}$ in the BE sub-domain by solving equation (4.26),

5. Apply:

$$\{\mathbf{u}^{I}_{F,n+1}\} = \{\mathbf{u}^{I}_{B,n}\} \tag{5.4}$$

The above solution is computed in two iterations: in the first iteration, the interface is fixed then the nodal forces on the interface for the FD sub-domain by Flac^{3D} program is computed. Afterward, the forces are transformed into nodal tractions, and finally the nodal displacement vector on the interface for the BE sub-domain is obtained by BEM. In this method the nodal displacement on the interface is applied on the FD sub-domain and the mechanical response in this domain is corrected over the Flac^{3D} algorithm in the second iteration. On the other hand, the computed nodal traction and displacement vectors on the interface are used to obtain the stress and displacement in the surrounding infinite or semi-infinite BEM sub-domain as a post processing step. The runtime (CPU) in this method for solving the spherical excavation problem, as shown in Table 5.15, is 71 % less than the first coupling method's CPU as shown in Table 5.4 and 59 % less than the optimum solution's CPU. The Flac^{3D} stress σ_z, seen in Figure 5.8 and Table 5.3 in Appendix b, p. 293, and displacements u_z solutions after applying this method are corrected and the error is significantly reduced (see Figures 5.52 and 5.53, and Table 5.16 in Appendix b, p.308-309). Relatively, accurate stress and displacement solutions in

the infinite sub-domain are computed (see Figures 5.54-5.56) if compared to the uncoupled Flac^{3D} solution (see Figure 5.8 and Table 5.3).

Table 5.15 Coupled Flac^{3D} -BEM vertical displacement (Second method/SDDM) values at point A, Material properties: E = 1000 MPa, ν = 0.0

E = 1000 Mpa ν = 0.0	R/a	Number of Zones	CPU [seconds]	uz Numerical [mm]	uz Exact [mm]	R.E. %
	2	1200	11.22	-1.0487	-0.9	16.52
	3	2592	25.00	-0.9650	-0.9	7.22
	4	6144	196.59	-0.9039	-0.9	0.44

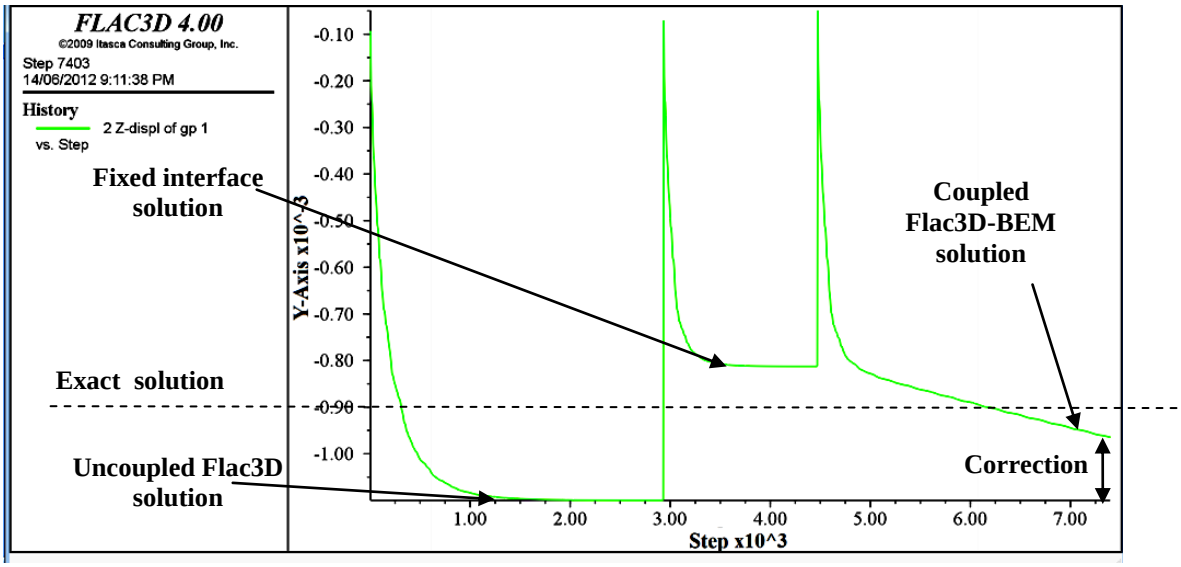


Figure 5.52 History of Flac3D and Coupled vertical displacement (Second method/SDDM) at point A, R/a =3, E = 1000 MPa, ν = 0.0, *(Y-Axis ≡ u_Z)*.

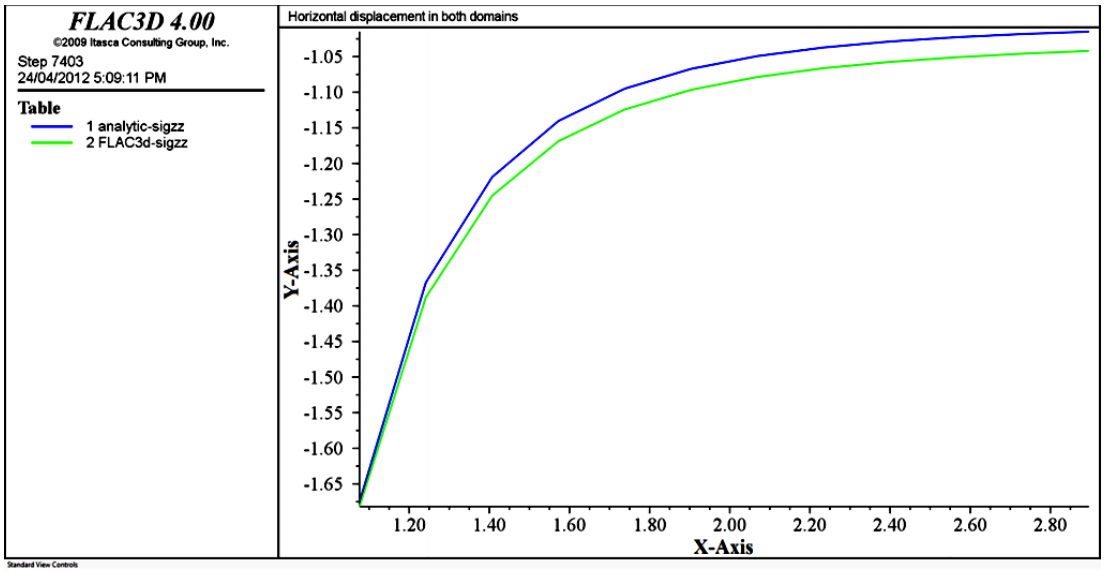


Figure 5.53 Coupled (Second method/SDDM) and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in the Flac3D sub-domain, R/a =3, E = 1000 & 2000 MPa, ν = 0.0, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

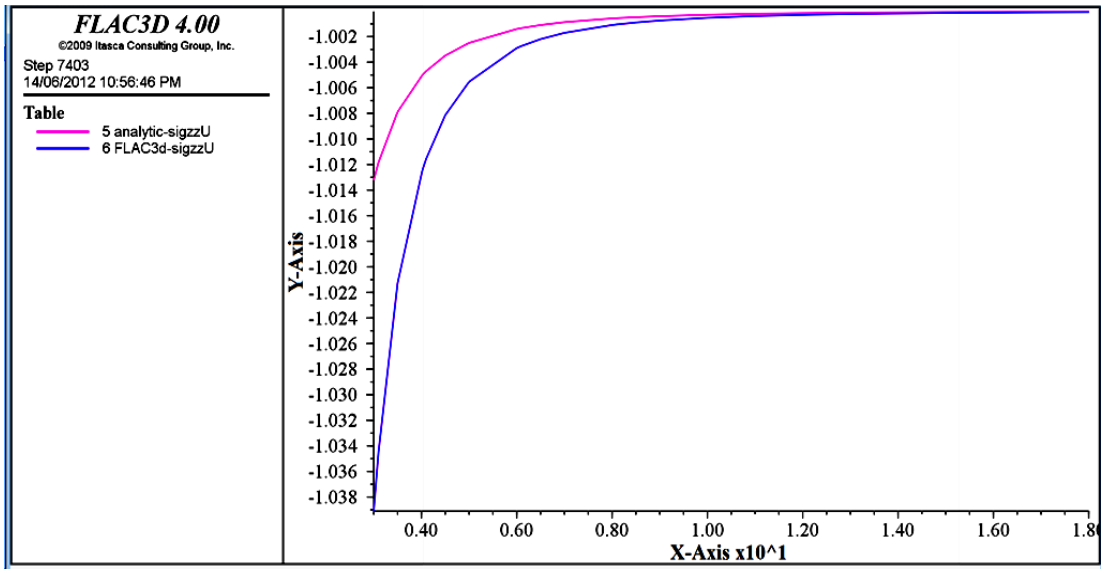


Figure 5.54 Coupled (Second Method/SDDM) and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in the BEM infinite sub-domain, R/a =3, E = 1000 & 2000 MPa, ν = 0.0, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

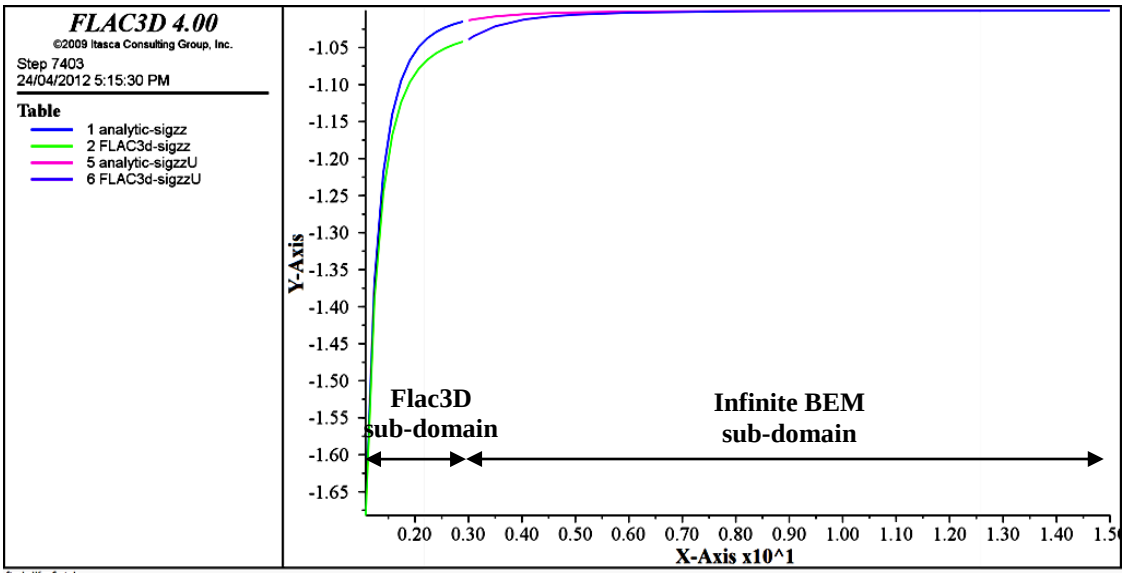


Figure 5.55 Coupled (Second Method/SDDM) and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in both Flac3D and BEM sub-domains, R/a =3, E = 1000 & 2000 MPa, ν = 0.0, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

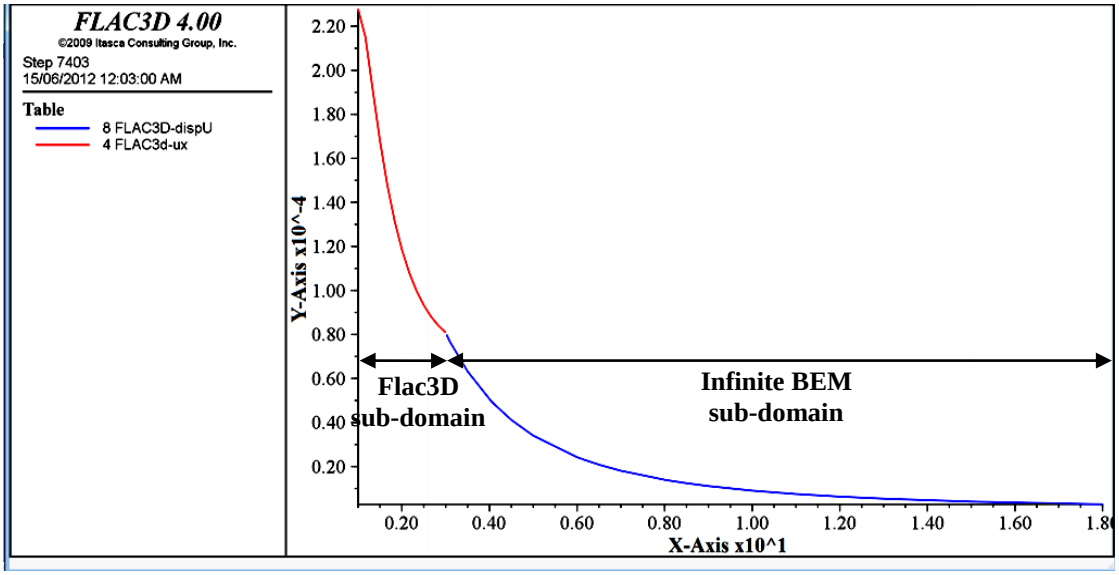


Figure 5.56 The coupled (Second Method/SDDM)horizontal displacement u_x in plane (z = 0) in both Flac3D and BEM sub-domains, R/a =3, E = 1000 MPa, ν = 0.0 , (X-Axis ≡ r) and (Y-Axis ≡ u_x).

The main disadvantages of this method are:

1. Less accurate solution is produced by the second coupling method, (see Figures 5.52-5.54) compared to the first coupling method, (see Figures 5.11 and 5.12).

2. The stress continuity and displacement compatibility across the interface are not fully developed in the second method, (see Figures 5.55 and 5.56).

3. Similar to the Flac^{3D} solution, the second method solution is truncation boundary position dependent, (see Table 5.16 in Appendix b, p.308-309).

4. Although, the second method solution is not affected by Young modulus material property value, (see Tables 5.16-5.17, Figures 5.53-5.55, and 5.57), it is dependent on Poisson's ratio υ values. The higher Poisson's ratio is the farther the truncation boundary should be positioned, (see Table 5.18.a and 5.18.b and Figures 5.58-5.60). See Table 5.18.b in Appendix b, p.310-311.

Table 5.17 Coupled Flac^{3D} -BEM vertical displacement (Second Method/SDDM) values at point A, Material properties: E = 2000 MPa, ν = 0.0

E = 2000 MPa ν = 0.0	R/a	Number of Zones	CPU [seconds]	uz Numerical [mm]	uz Exact (mm)	R.E. %
	2	1200	11.81	-0.5244	-0.45	16.52
	3	2592	25.14	-0.48	-0.45	7.22
	4	6144	196.59	-0.45	-0.45	0.46

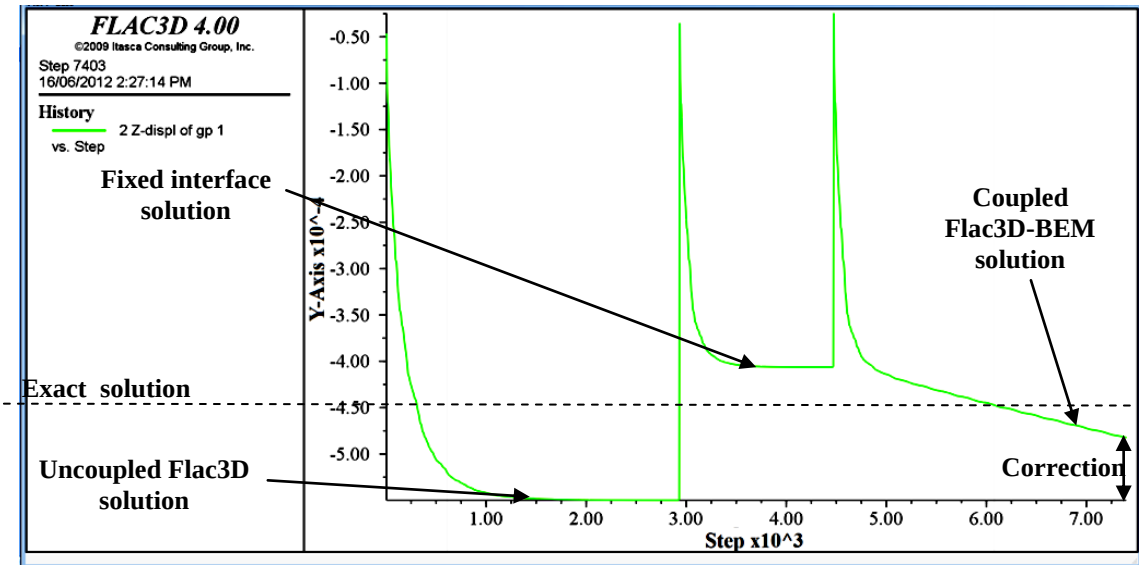


Figure 5.57 History of Flac3D and Coupled vertical displacement (Second Method/SDDM) at point A, R/a =3, E = 2000 MPa, ν = 0.0, *(Y-Axis ≡ u_Z)*

Table 5.18.a Coupled Flac^{3D} -BEM vertical displacement (Second Method/SDDM) values at point A, Material properties: E = 1000 MPa, ν = 0.3

	R/a	Number of Zones	CPU [seconds]	uz Numerical [mm]	R.E. %
E = 1000 MPa ν = 0.3	2	6144	86.39	-1.1894	18.94
	3	12000	162.65	-1.0902	9.02
	4	20736	335.31	-1.0403	4.03
	6	32928	673.22	-1.0012	0.12

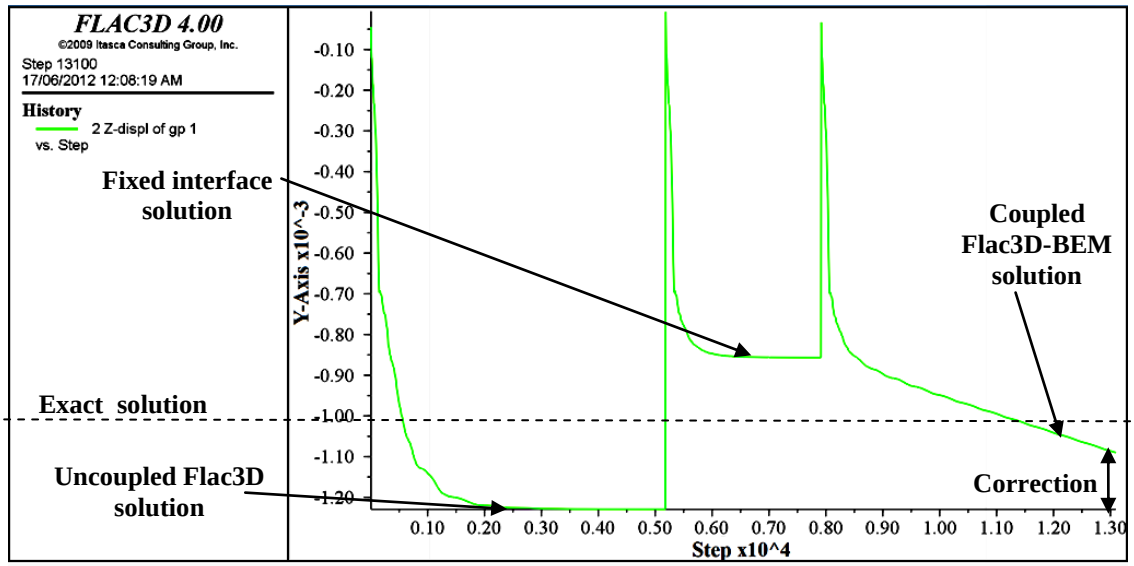


Figure 5.58 History of Flac3D and Coupled vertical displacement (second Method/SDDM) at point A, R/a =3, E = 1000 MPa, ν = 0.3, *(Y-Axis ≡ u_Z)*

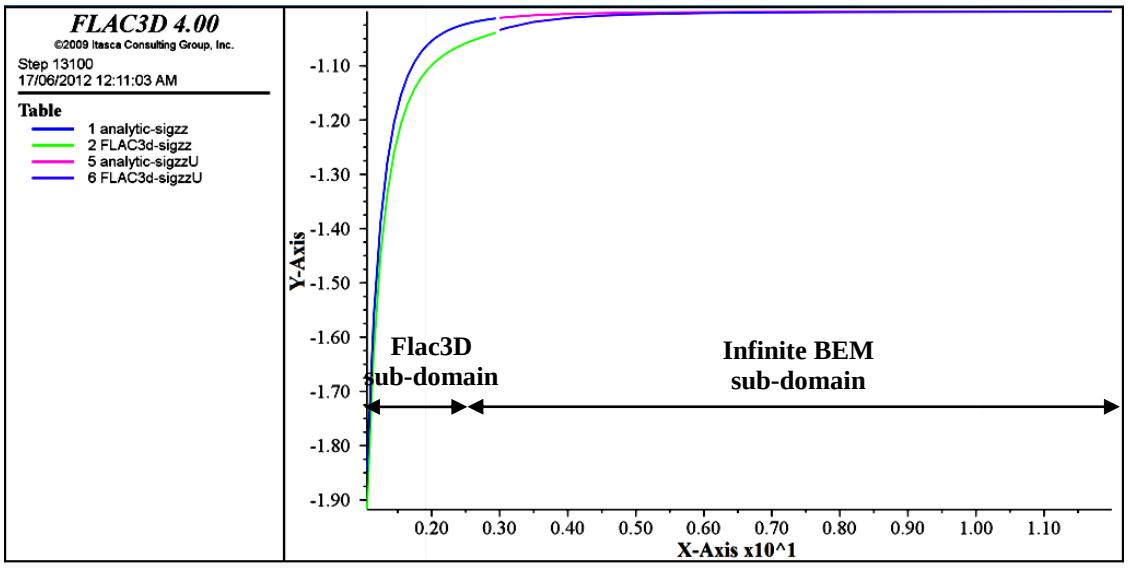


Figure 5.59 Coupled (Second method) and exact vertical stress (σ_z) in plane (z = 0) as a function of (r) in both Flac3D and BEM sub-domains, R/a =3, E = 1000 MPa, ν = 0.3, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z).*

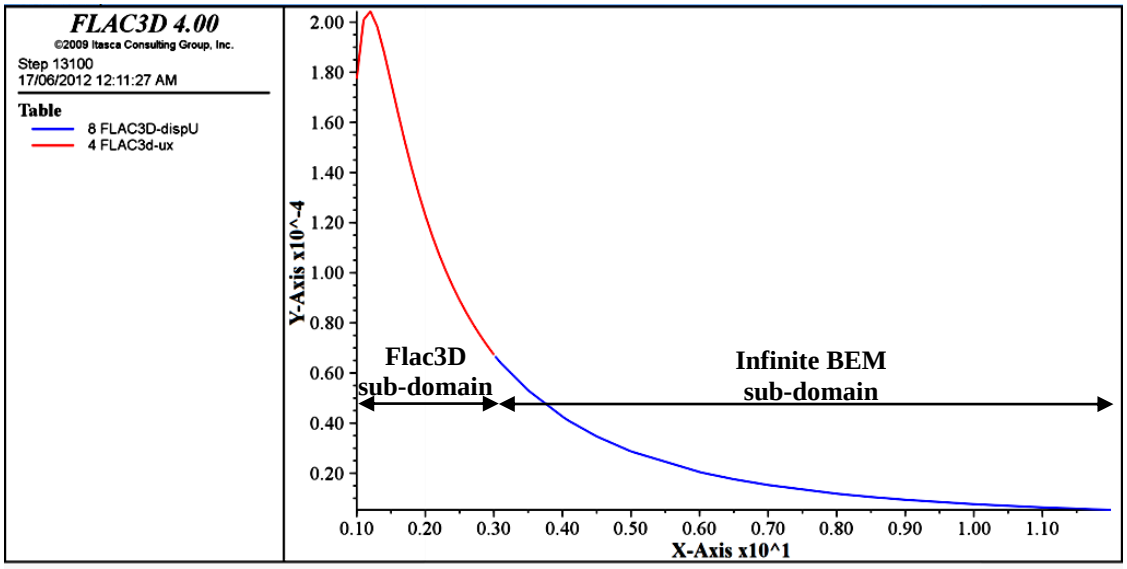


Figure 5.60 The coupled (Second Method/SDDM) horizontal displacement u_x in plane (z = 0) in both Flac3D and BEM sub-domains, R/a =3, E = 1000 MPa, ν = 0.3, (X-Axis ≡ r) and (Y-Axis ≡ u_x).

5.3.2 Square Uniform Load on 3D Semi-infinite Medium

Timoshenko and Goodier [125] presented the analytical solution for the vertical displacement u_z under the center and the corner of a square uniform load applied on z = 0 plane of a homogeneous half space domain. The displacement u_z at point A (see Figure 5.60) in the center of a square of side length 2a loaded by a uniform load p at z = 0 is given by:

$$u_z = \frac{8}{\pi}\ln(\sqrt{2}+1)\frac{(1-\nu^2)pa}{E} \tag{5.5}$$

The displacement under the corners (point B) at z = 0 is half of the center displacement.

Young modulus is given the value E = 10000 kN/m^2 as an example, Poisson's ratio = 0, a = 1 m and p = 100 kN/m^2.

5.3.2.1 Uncoupled BEM Solution

A. Uncoupled BEM using Mindlin's Fundamental Solutions (UMFS)

The BE problem solution is approximated numerically in two methods: in UMFS, the first method, the BIE (2.72), in its discretized formulation (the system of equations 2.60), is solved using Mindlin's fundamental solutions as defined in equation (2.71). In this method only the loaded part of the free traction surface is considered as a boundary according to equation (2.72). The square load is divided into four quadrilateral 8-node elements, and because of the symmetry of the problem, only one element is used to do the analysis (Figure 5.60-a).

Table 5.19 Vertical displacement u_z under the loaded square area [m]

Point	A [Center]	B [Corner]
Exact	0.02244399	0.01122200
BEM [Mindlin]	0.02244403	0.01122201
R.E.%	0.000160	0.000115
BEM [Infinite 6-node Elements]	0.02242900	0.01136200
R.E.%	-0.066807	1.247576
BEM [Infinite 4-node Elements]	0.022178	0.011838
R.E.%	-1.185146	5.489245

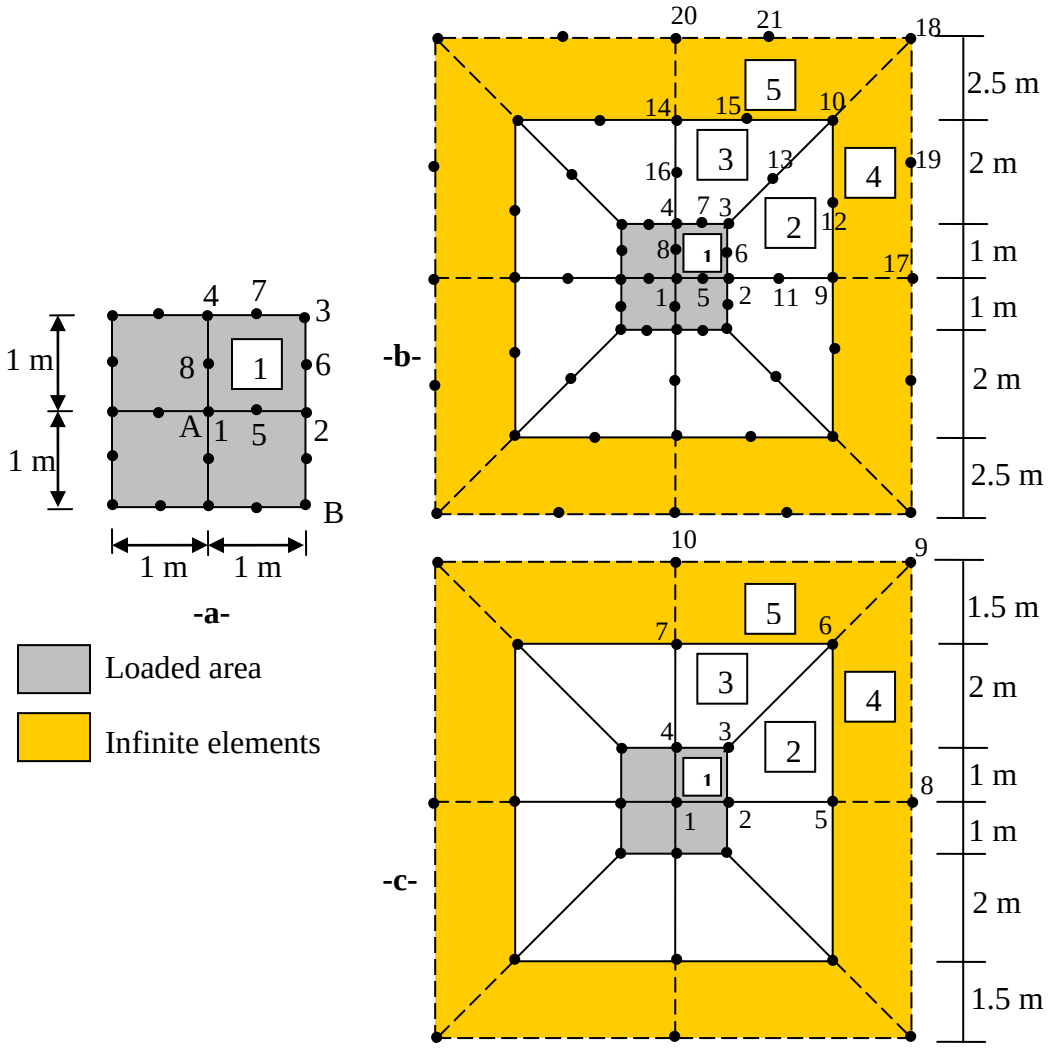


Figure 5.60 a-Boundary mesh with 8-node finite BEs for Mindlin's method
b- Boundary mesh with 6-node infinite and 8-node finite BEs
c- Boundary mesh with 4-node infinite and 4-node finite BEs

The vertical displacement u_z at points A and B in Mindlin method is in perfect agreement with the analytical solution as shown in Table 5.19. Holl [126] in 1940 derived the six stress components under the corner of a rectangular uniform load applied on z = 0 plane of a homogeneous half space domain (Poission's ratio = 0.5) and presented by Poulos and Davis [127] as the following:

$$\sigma_z = \frac{p}{2\pi}\left[\tan^{-1}\frac{lb}{zR_3} + \frac{lbz}{R_3}\left(\frac{1}{R_1^2} + \frac{1}{R_2^2}\right)\right]$$

$$\sigma_x = \frac{p}{2\pi}\left[\tan^{-1}\frac{lb}{zR_3} - \frac{lbz}{R_1^2 R_3}\right]$$

$$\sigma_y = \frac{p}{2\pi}\left[\tan^{-1}\frac{lb}{zR_3} - \frac{lbz}{R_2^2 R_3}\right] \qquad (5.6)$$

$$\sigma_{xz} = \frac{p}{2\pi}\left[\frac{b}{R_2} - \frac{z^2 b}{R_1^2 R_3}\right]$$

$$\sigma_{yz} = \frac{p}{2\pi}\left[\frac{l}{R_1} - \frac{z^2 l}{R_2^2 R_3}\right]$$

$$\sigma_{xy} = \frac{p}{2\pi}\left[1 + \frac{z}{R_3} - z\left(\frac{1}{R_1} - \frac{1}{R_2}\right)\right]$$

Where: $R_1 = \left(l^2 + z^2\right)^{1/2}$

$R_2 = \left(b^2 + z^2\right)^{1/2}$

$R_3 = \left(l^2 + b^2 + z^2\right)^{1/2}$

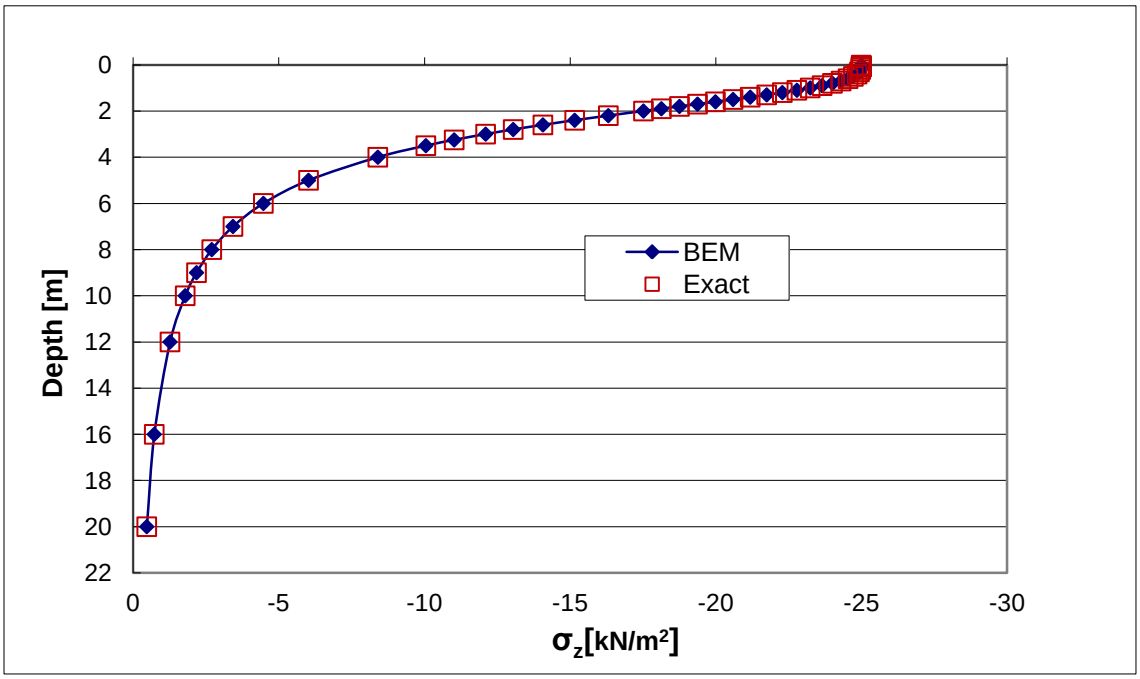
l and b are the rectangle length and width, respectively.

The author modified the x and y components of the normal stress to become for elastic medium of Poisson's ratio equals to zero and square load (l = b):

$$\sigma_x = \frac{p}{4\pi}\left[\tan^{-1}\frac{l^2}{zR_3} - \frac{2l^2 z}{R_1^2 R_3}\right] \qquad (5.7)$$
$$\sigma_x = \sigma_y$$

The modification is tested against two theoretical and numerical criteria, as will be demonstrated. The first criterion, the Mindlin numerical solution for the vertical displacement under the center and the corner of the square load was almost 100% accurate as shown in Table 5.19. On the other hand, the stress components σ_z, τ_{xz} and τ_{yz} under the corner and σ_z under the center also obtained by the same method, are in perfect agreement with the analytical solution in equations (5.6), see Figures 5.61, 5.63 (see Appendix c, p.312) and 5.65 (see Appendix c, p.314). Moreover, the perfect agreement between the Mindlin numerical solution and the modified analytical solution for both stress components σ_x and σ_y under points A and B is a strong indication that the author's modification is accurate, see Figures 5.62 and 5.66 (see Appendix c p.315). These stress components are computed inside the semi-infinite domain as a post processing step. Equation (2.67) is solved with **S** and **R** fundamental solutions. The complementary part of these tensors $\mathbf{S^c}$ and $\mathbf{R^c}$ in equations (2.85), based on Mindlin fundamental solutions, have been derived by the author with the assistance of the Matlab 7.1 program. Afterward, the author coded tensors $\mathbf{S^c}$ and $\mathbf{R^c}$ for computer computations into FORTRAN subroutines and C^{++} functions as well (see section 2.2.10 and Appendix f, pp. 352-372).

a.



b.

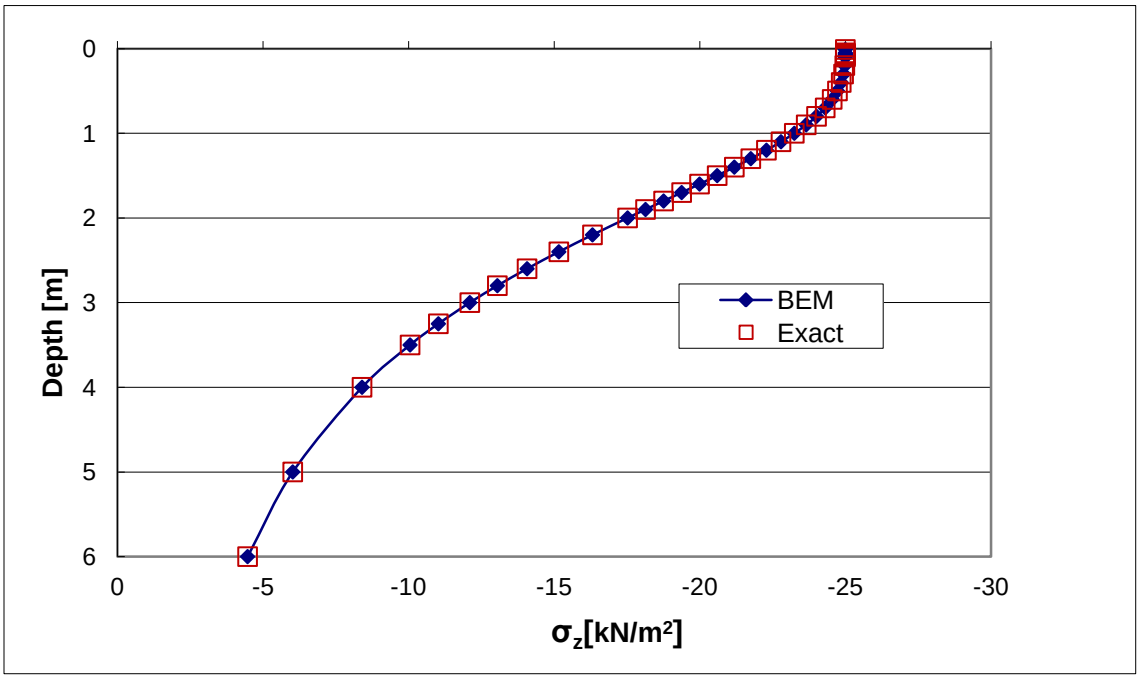
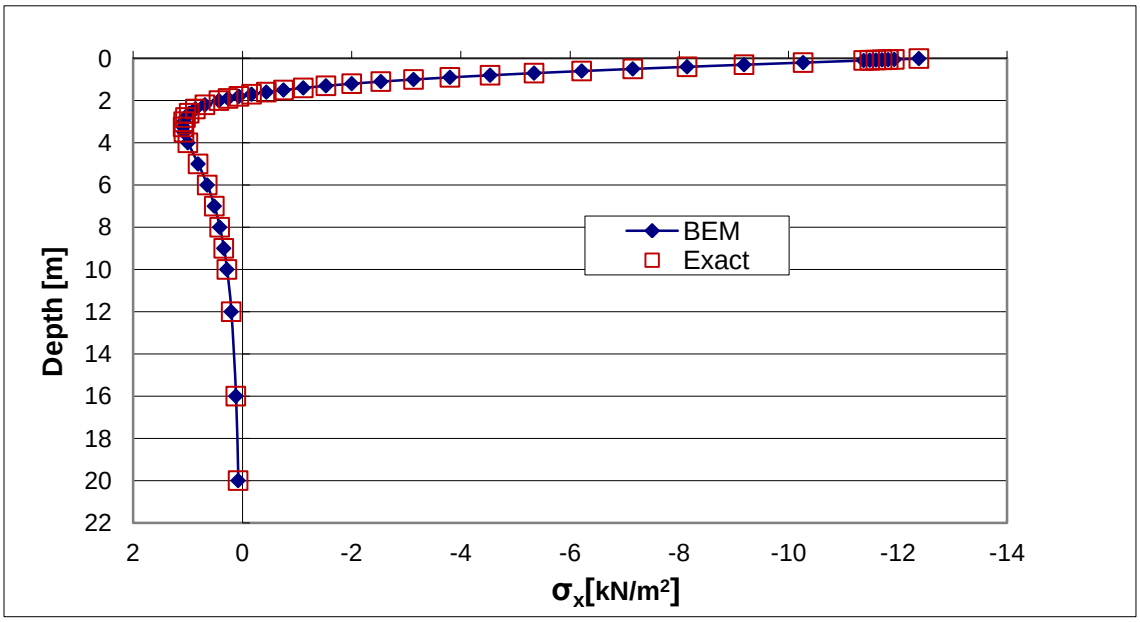


Figure 5.61 a. & b. The vertical stress σ_z under the corner of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0$].

a.



b.

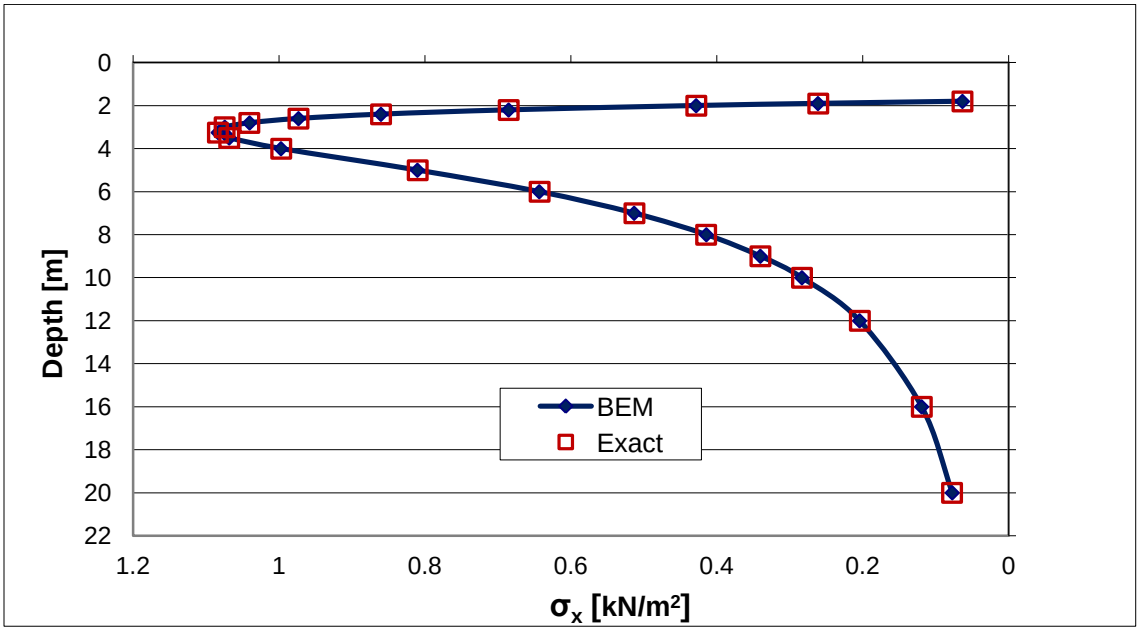


Figure 5.62 a. & b. The horizontal stress σ_x under the corner of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0$].

Harr [127] obtained the analytical solution for the vertical displacement u_z under the corner of a uniformly loaded rectangular area:

$$u_z = \frac{pb}{E}(1-\upsilon^2)(A - \frac{1-2\upsilon}{1-\upsilon}B) \qquad (5.8)$$

Where: $A = \frac{1}{2\pi}(\ln \frac{\sqrt{1+m_1^2+n_1^2}+m_1}{\sqrt{1+m_1^2+n_1^2}-m_1} + m_1 \ln \frac{\sqrt{1+m_1^2+n_1^2}+1}{\sqrt{1+m_1^2+n_1^2}-1}$

$$B = \frac{n_1}{2\pi}\tan^{-1}\frac{m_1}{n_1\sqrt{1+m_1^2+n_1^2}},\ m_1 = \frac{l}{b} \text{ and } n_1 = \frac{z}{b}$$

The perfect agreement between Mindlin's BEM solution and Harr exact solution for the vertical displacement u_z under the corner and in the center of the loaded square is another indication to the convergence of this numerical solution, see Figures 5.67 and 5.68.

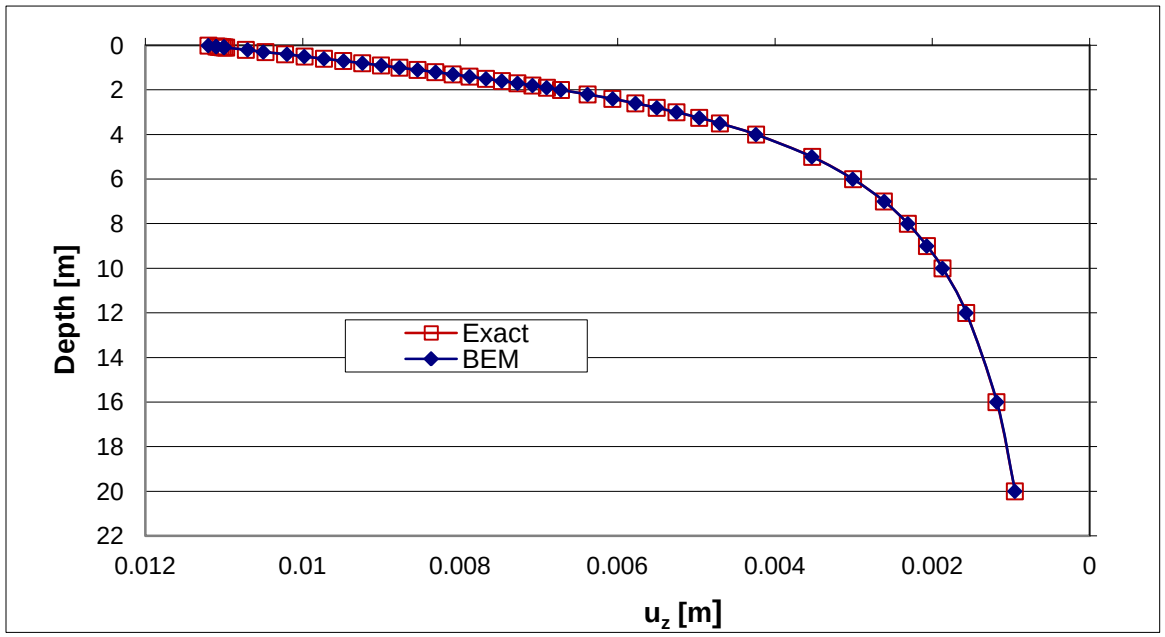


Figure 5.67 Vertical displacement u_z under the corner of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0$].

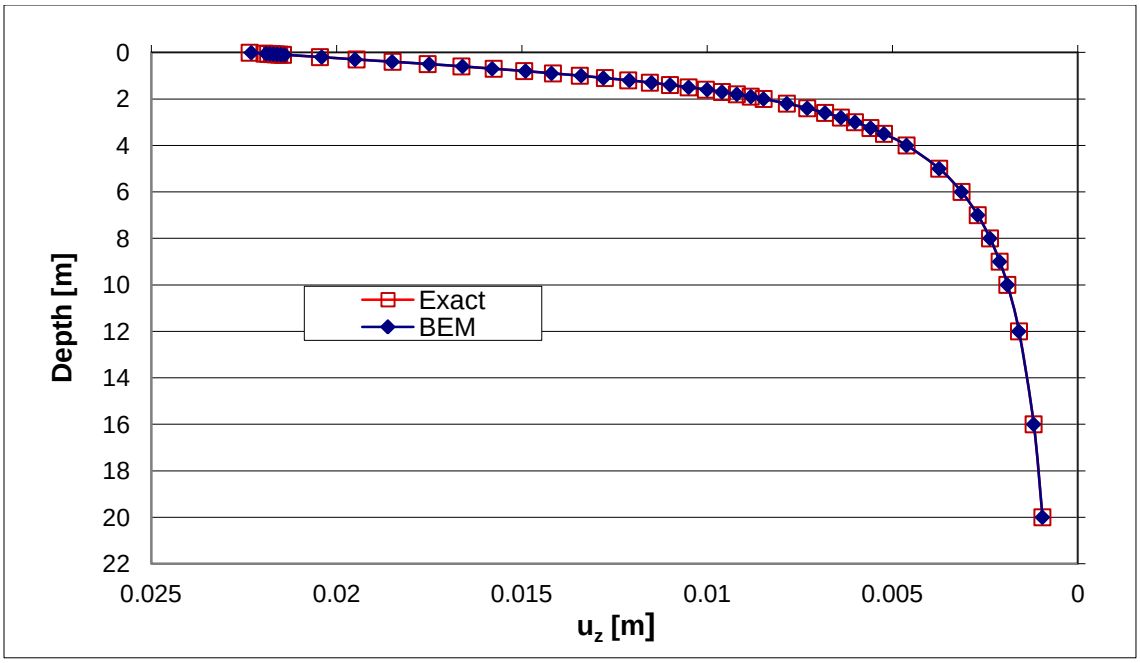


Figure 5.68 Vertical displacement u_z under the center of a uniform square load [Analytical and Mindlin's BEM solutions, υ = 0].

In the second criterion, the modified σ_x and σ_y are tested according to theoretical criteria. Giroud [127] presented the following normal stress components under the corner of uniform load on rectangular area:

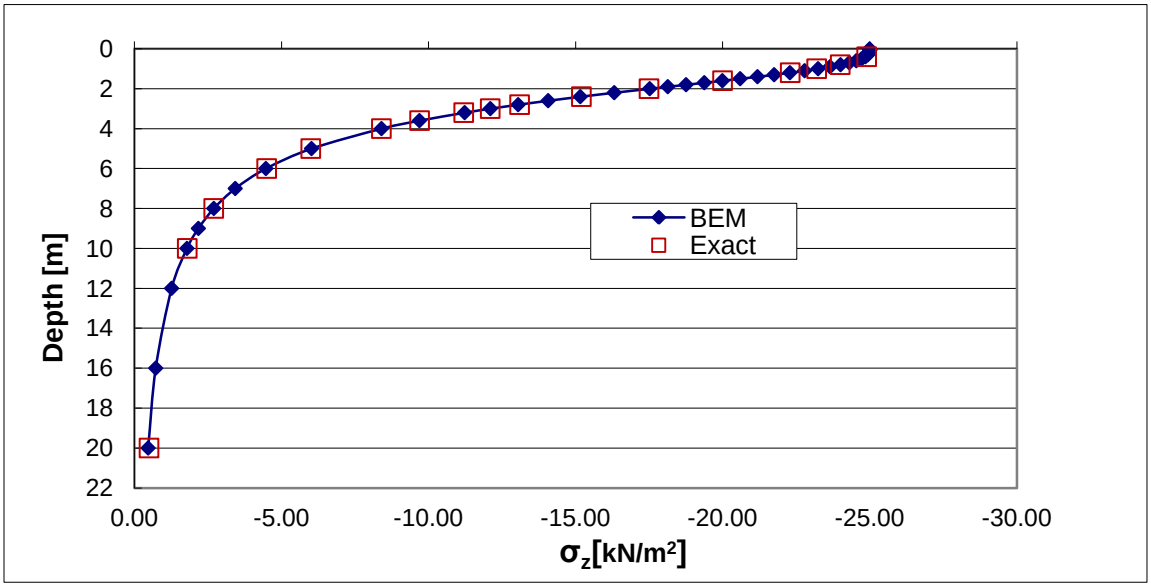
$$\begin{aligned}\sigma_z &= pK_0 \\ \sigma_x &= p\left[K_2 - (1-2\upsilon)K_2^1\right] \\ \sigma_y &= p\left[L_2 - (1-2\upsilon)L_2^1\right]\end{aligned} \qquad (5.9)$$

Where: K_0, K_2, K_2^1, L_2 and L_2^1 are influence factors tabulated in reference [126], pp. 55-57.

To verify the modification, Poisson's ratio is given a zero value and substituted into equation (5.9). The resulted σ_x and σ_y values are the same compared to the values reproduced using equations (5.7). The numerical BEM Mindlin solution is also verified for different Poisson's ratio value (υ = 0.3). A perfect agreement between the numerical and Giroud exact solution for the normal stress components under the corner and the center of the loaded area, see Figures 5.69, 5.70 and Figures 5.73, 5.74 (both are in Appendix c, pp.317-318). It should be noticed

that Poisson's ratio (υ) has a minor effect over σ_z values. The vertical stress component σ_z in both equations (5.9) and (5.6) has almost the same values for different given values to Poisson's ratio.

a.



b.

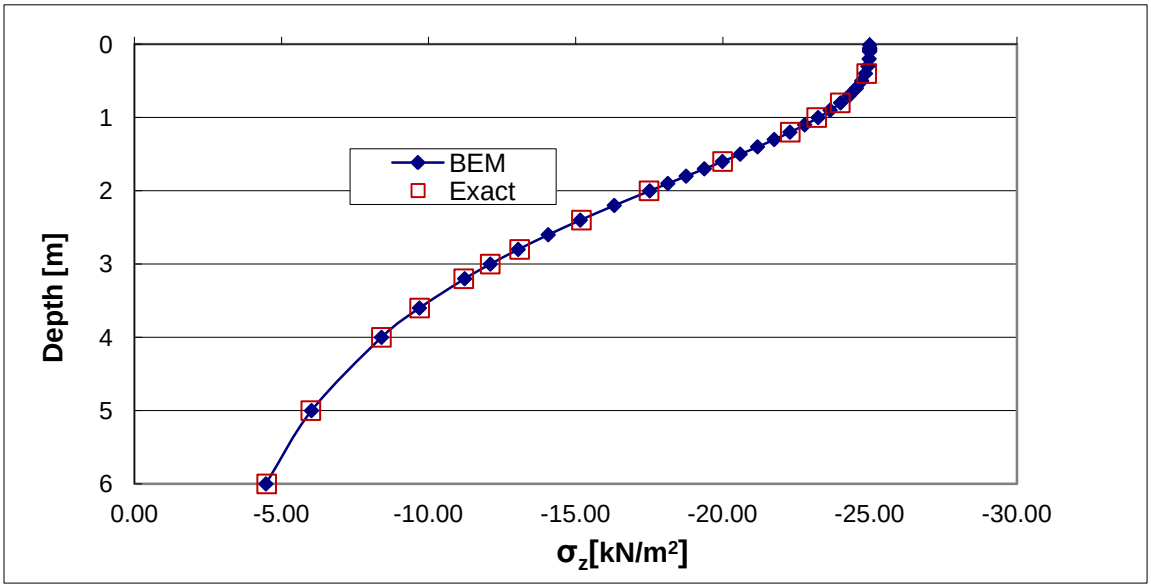
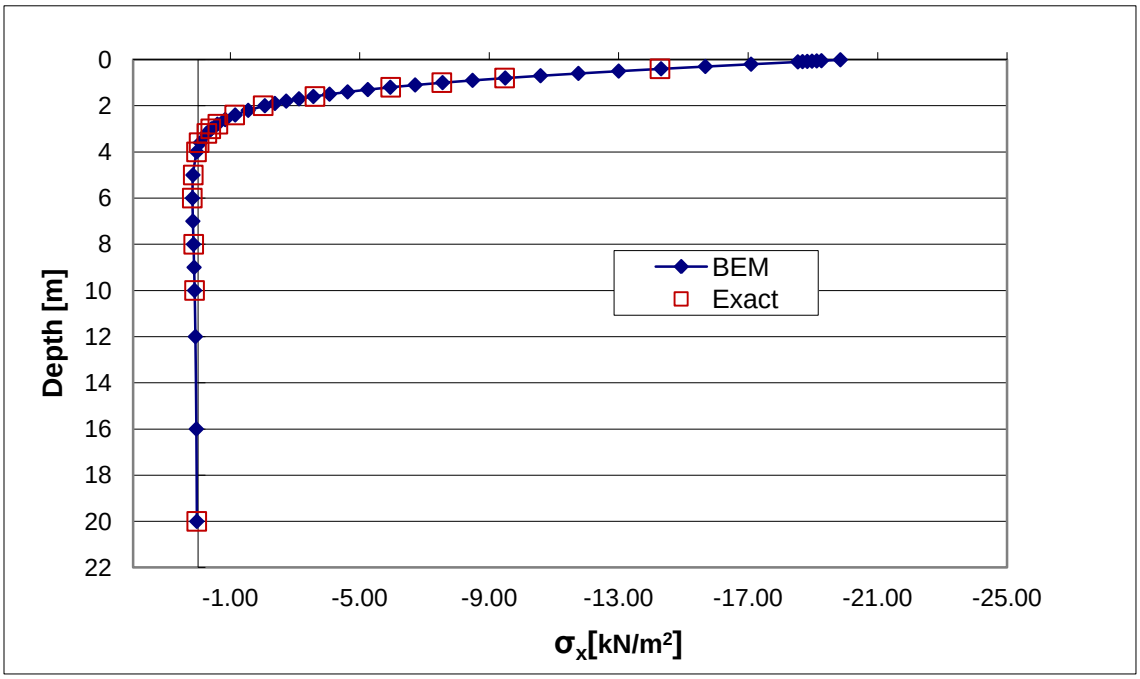


Figure 5.69 a. & b. The vertical stress σ_z under the corner of a uniform square load [Analytical and Mindlin's BEM solutions, υ = 0.3].

a.



b.

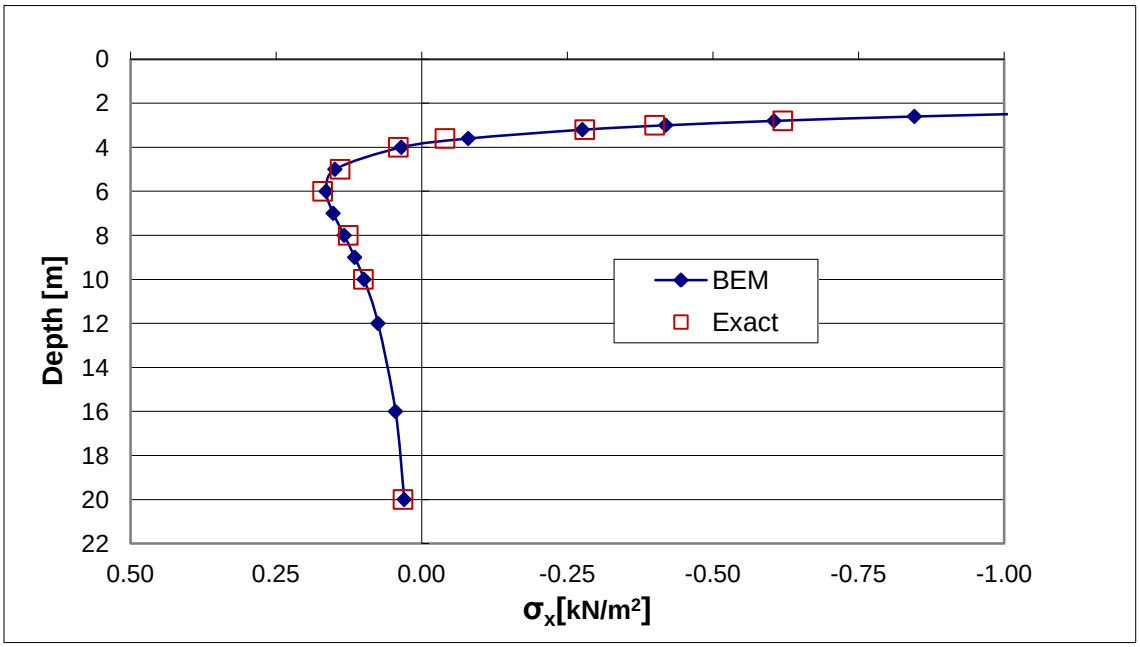


Figure 5.70 a. & b. The horizontal stress σ_x under the corner of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0.3$].

Mindlin's BEM solution converges to Harr's exact solution for the vertical displacement u_z under the corner and the center of the loaded square in the case of Poisson's ratio value (υ = 0.3). The effect of the Poisson ratio is significantly incorporated in this numerical solution, as shown in Figures 5.75, 5.76.

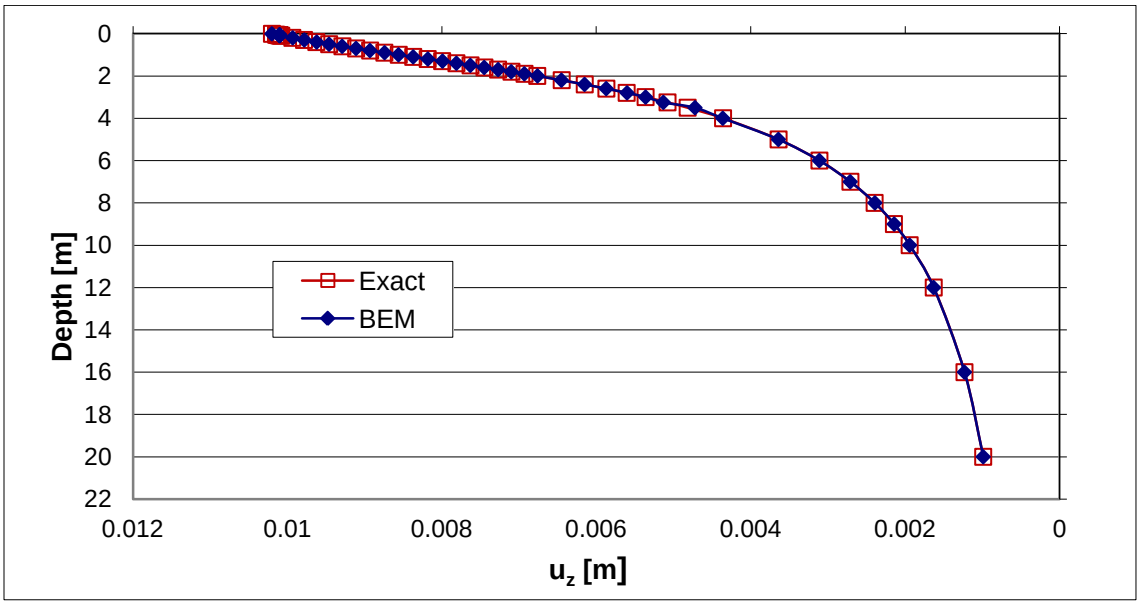


Figure 5.75 Vertical displacement u_z under the corner of a uniform square load [Analytical and Mindlin's BEM solutions, υ = 0.3].

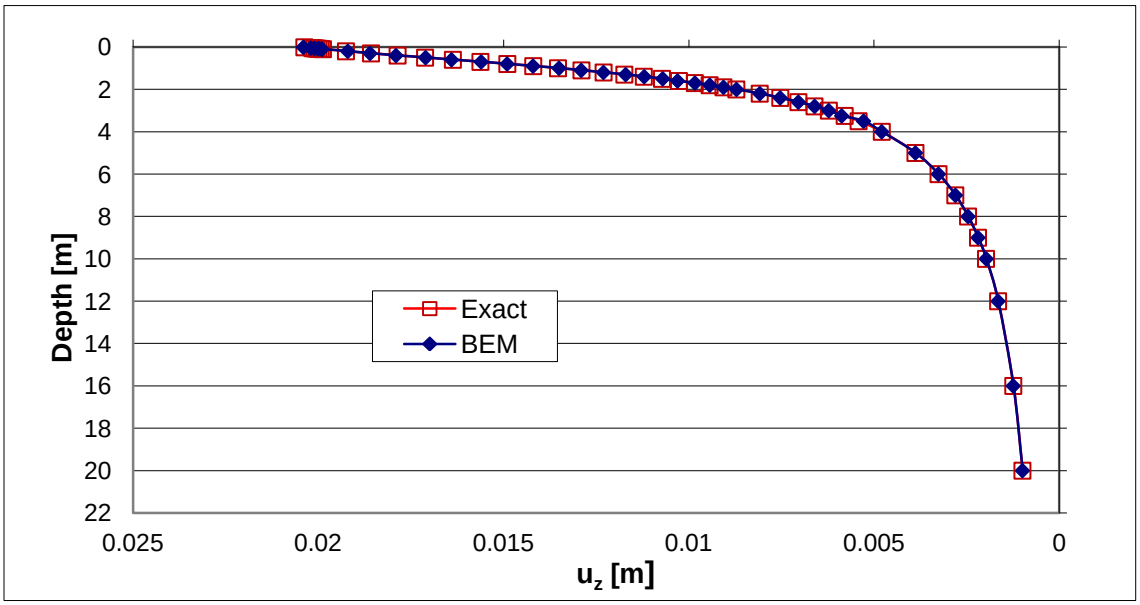


Figure 5.76 Vertical displacement u_z under the center of a uniform square load [Analytical and Mindlin's BEM solutions, υ = 0.3].

B. Uncoupled BEM using Infinite Boundary Elements (UIBE)

The UIBE, as explained earlier in section 2.2.8.2, solves the BIE (2.37) with Kelvin's fundamental solution. The ground surface (plane z = 0) in this method is subdivided into isoparametric finite BEs and infinite BEs. The boundary as shown in Figures 5.60-b and 5.60-c is discretized into: b. Three quadrilateral 8-node elements and two infinite 6-node boundary elements; c. Three quadrilateral 4-node elements and two infinite 4-node boundary elements.

The number of elements is reduced from 20 to 5 because of the problem symmetry.

The vertical displacement u_z solution for the same problem data (E = 10000 kN/m^2, υ = 0, a = 1 m and p = 100 kN/m^2) at points A and B computed numerically with the UIBE method using infinite elements is not as accurate as Mindlin's method (UMFS). Nevertheless, the results are accurate within one to five percent relative error with the exact solution (see Table 5.19). This is important for cases where we have to use infinite elements anyway as explained in section 2.2.8.2. Although, u_z solution at A and B in mesh (b) is more accurate than mesh (c), the stress components' accuracy inside the semi-infinite domain under the corner and the center of the loaded for both meshes are not significantly different. However, the vertical displacement under the load corner and center inside the semi-infinite domain is not as accurate as the Mindlin's (UMFS method) values (see Figures 5.77 and 5.78). Mesh size should be finer to achieve accurate stress solution in the region close to the loading level as Figures 5.79 up to 5.83 illustrate (see Appendix c, pp.319-321). Larger mesh size makes the cost of the solution more expensive in terms of runtime (CPU).

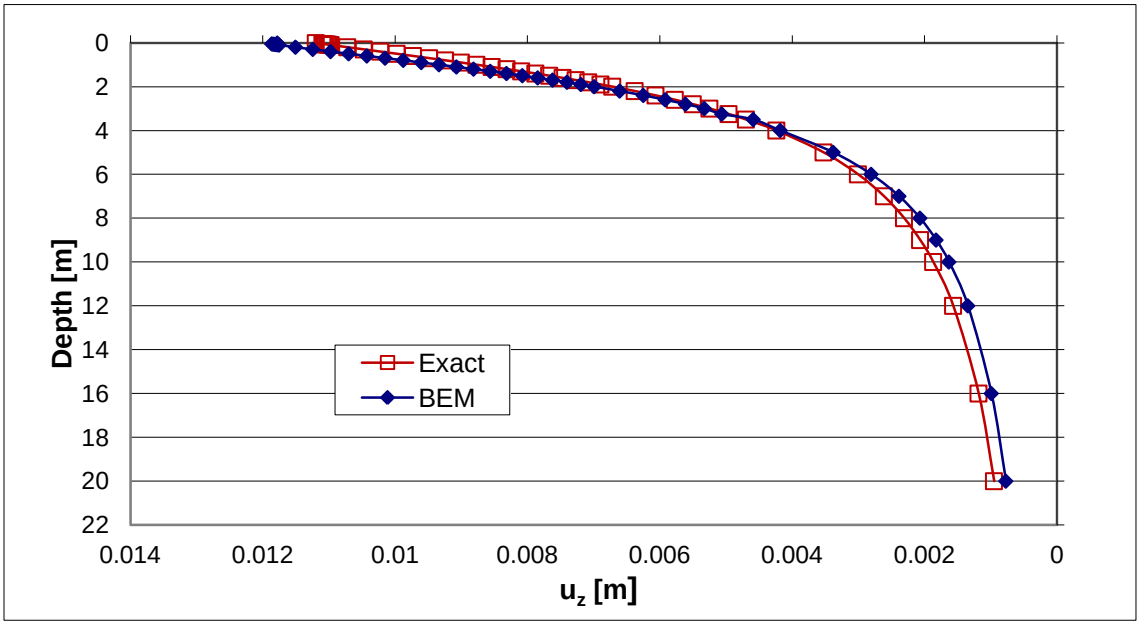


Figure 5.77 Vertical displacement u_z under the corner of a uniform square load [Analytical and Infinite BE solutions, $\upsilon = 0$].

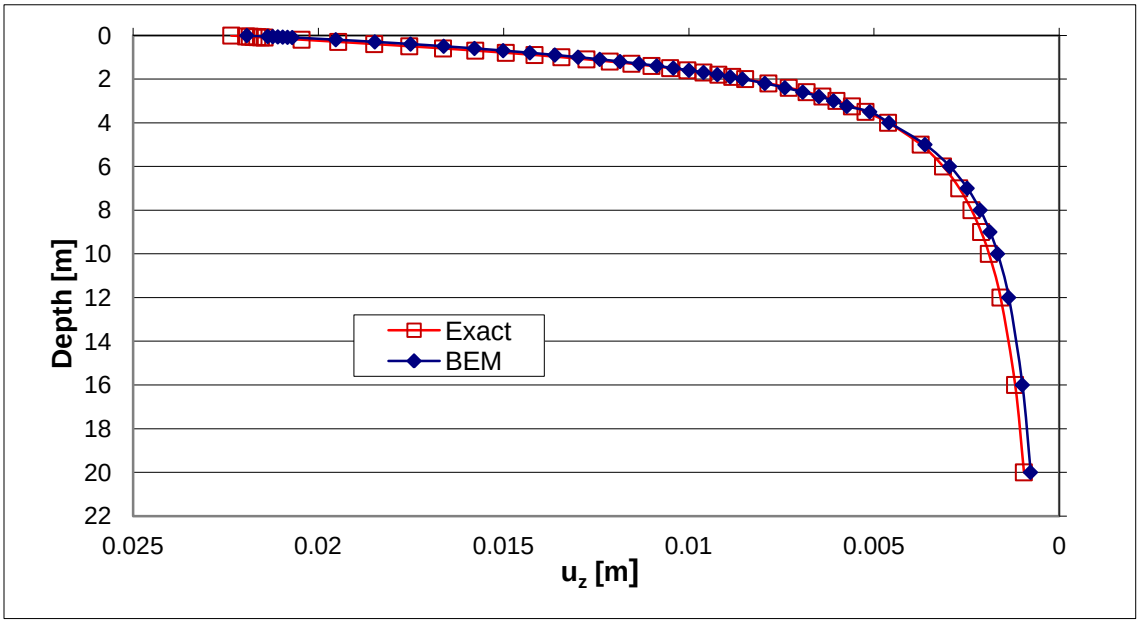


Figure 5.78 Vertical displacement u_z under the center of a uniform square load [Analytical and Infinite BE solutions, $\upsilon = 0$].

5.3.2.2 Uncoupled Flac^{3D} Solution (v = 0)

The Flac^{3D} model used to solve the same problem is of radius R =3 m or ratio (R/a) = 3 and mesh size or No. of zones = 2048. Material properties are E = 10000 kN/m^2 and Poisson's ratio = 0. Because of the symmetrical nature of the problem in terms of xz and yz planes, only (1/4) of the loaded semi-sphere is modeled. The Flac^{3D} model truncation boundary is fixed in the vertical direction, but it is freed in the x and y directions (see Figure 5.84).

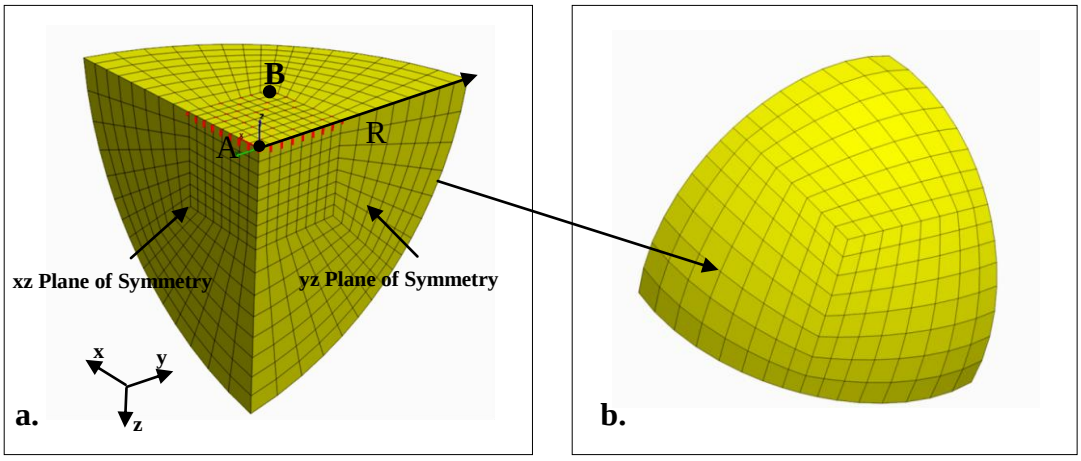


Figure 5.84 a. Flac3D Model of ratio (R/a = 3)
b. Flac3D-BEM sub-domains' interface

Because of the partially fixed boundary, the vertical displacement u_z at the center and the corner of the load on plane z = 0 (see Table 5.20), the vertical and horizontal stress under the load center (see Figures 5.85 and 5.86 and Table 5.21 in Appendix c, p.322), the vertical, horizontal and shear stress under the load corner (see Figures 5.88-5.90), the vertical displacement u_z under the load center (see Figure 5.87 and Table 5.22 in Appendix c p.322), and the vertical displacement u_z under the load corner (see Figure 5.91 and Table 5.23 in Appendix c p.323) are all significantly underestimated.

Table 5.20 The vertical displacement u_z under a loaded square area obtained by two methods, ν = 0

Uncoupled Flac^{3D} Solution [coarse mesh]						
Point	**R/a**	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [Center]	3	0.022443994	0.01729818	-22.93	14.527	2048
B [Corner]		0.011221997	0.00653903	-41.73		
Uncoupled Flac^{3D} Solution [fine mesh]						
Point	**R/a**	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [Center]	35	0.022443994	0.021918	-2.34	217.03	34496
B [Corner]		0.011221997	0.01088758	-2.98		
Coupled BEM-Flac^{3D} Solution [First Method]						
Point	**R/a**	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [Center]	3	0.02244399	0.02223186	-0.95	139.7	2048
B [Corner]		0.01122200	0.01132729	0.94		

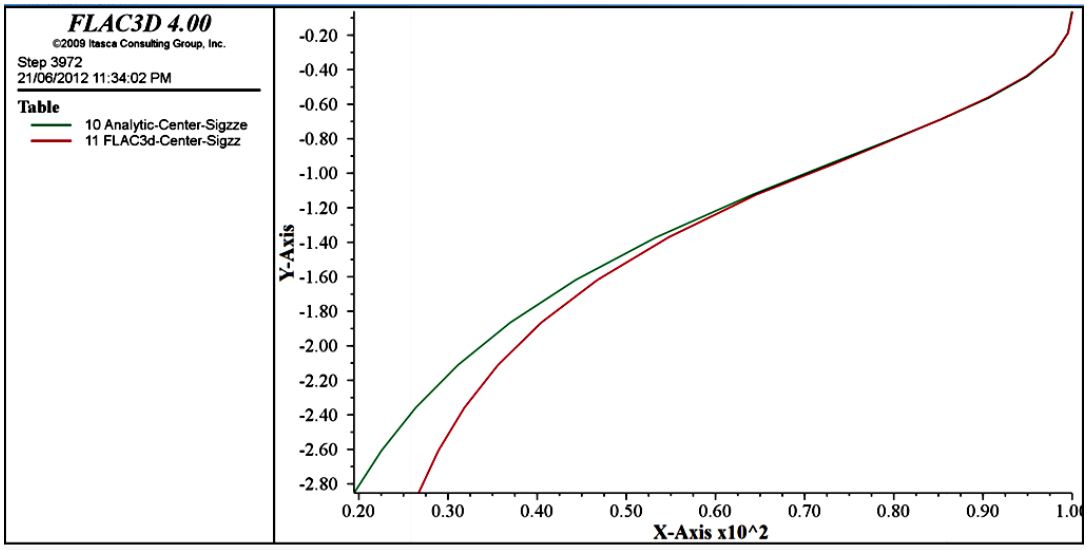


Figure 5.85 Uncoupled Flac3D and exact vertical stress (σ_z) solution under the center of a uniform square load, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

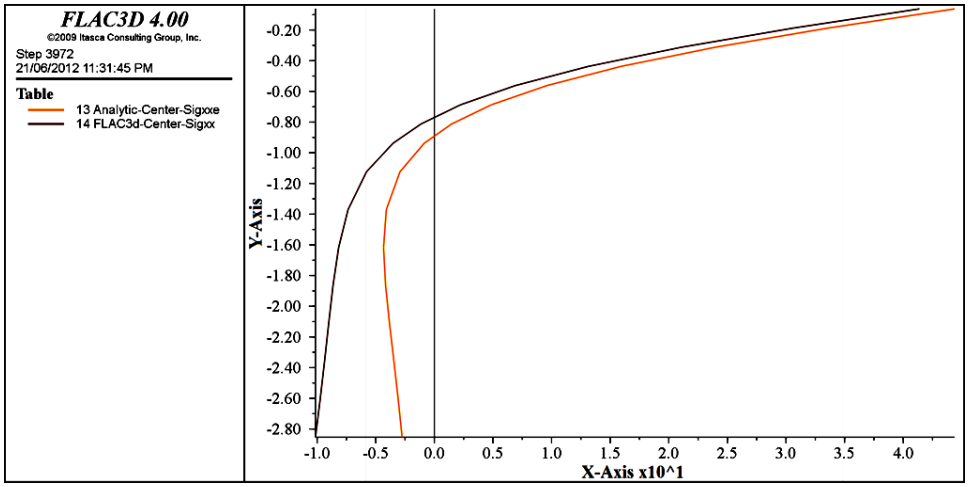


Figure 5.86 Uncoupled Flac3D and exact horizontal stress (σ_x) solution under the center of a uniform square load, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

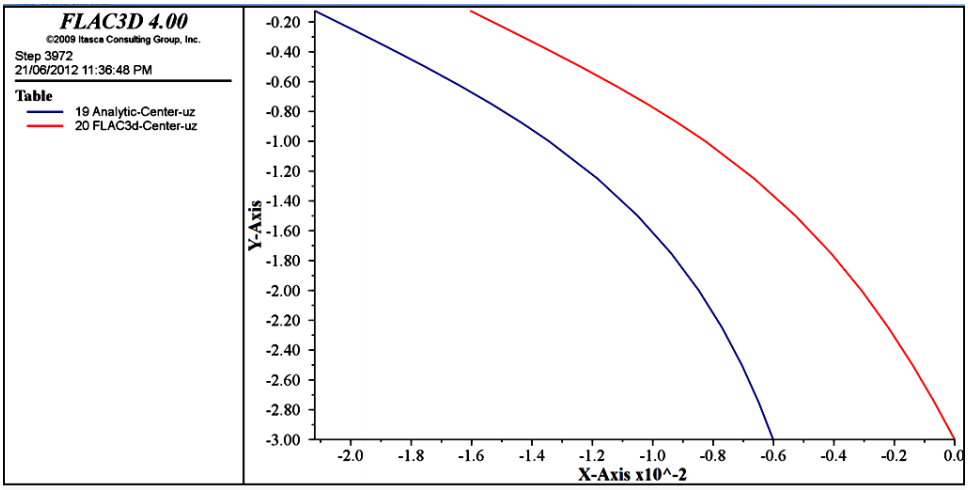


Figure 5.87 Uncoupled Flac3D and exact vertical displacement (u_z) solution under the center of a uniform square load, ν = 0.0,*(Y-Axis ≡ Depth) and (X-Axis ≡ u_z).*

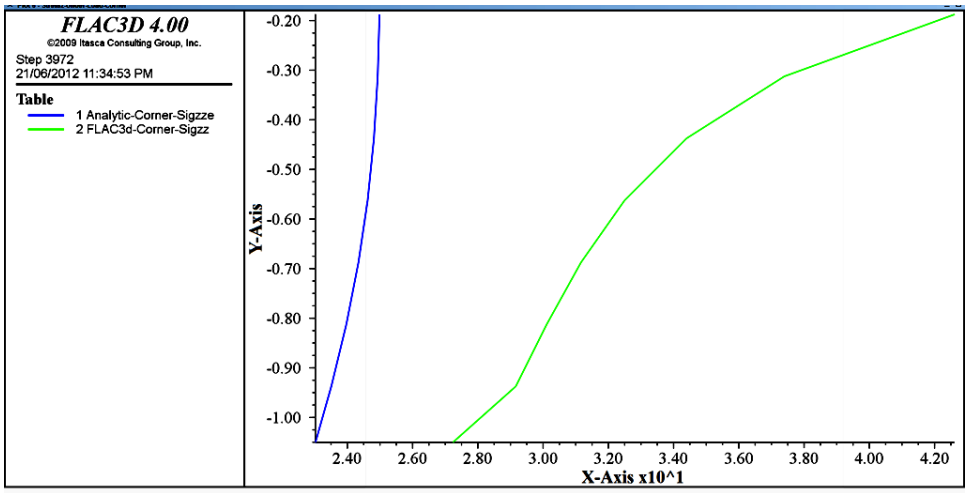


Figure 5.88 Uncoupled Flac3D and exact vertical stress (σ_z) solution under the corner of a uniform square load, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

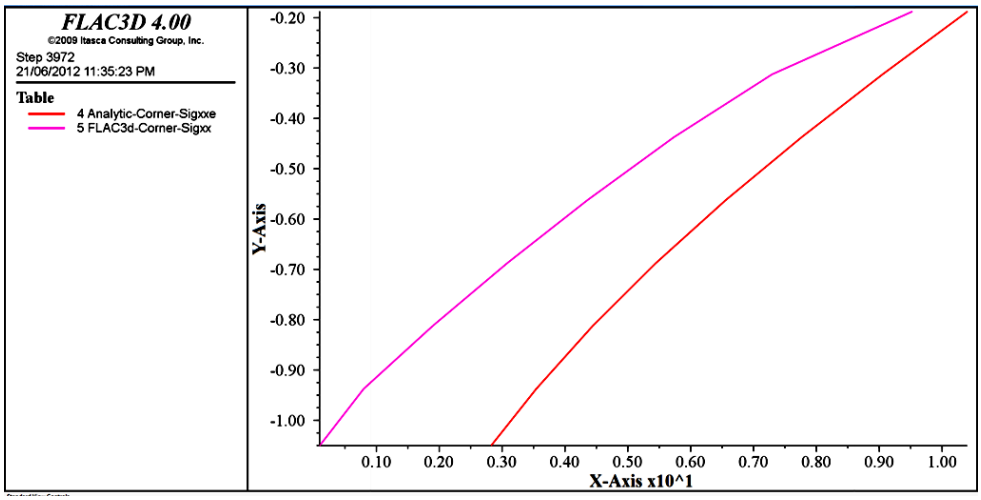


Figure 5.89 Uncoupled Flac3D and exact horizontal stress (σ_x) solution under the corner of a uniform square load, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x)*.

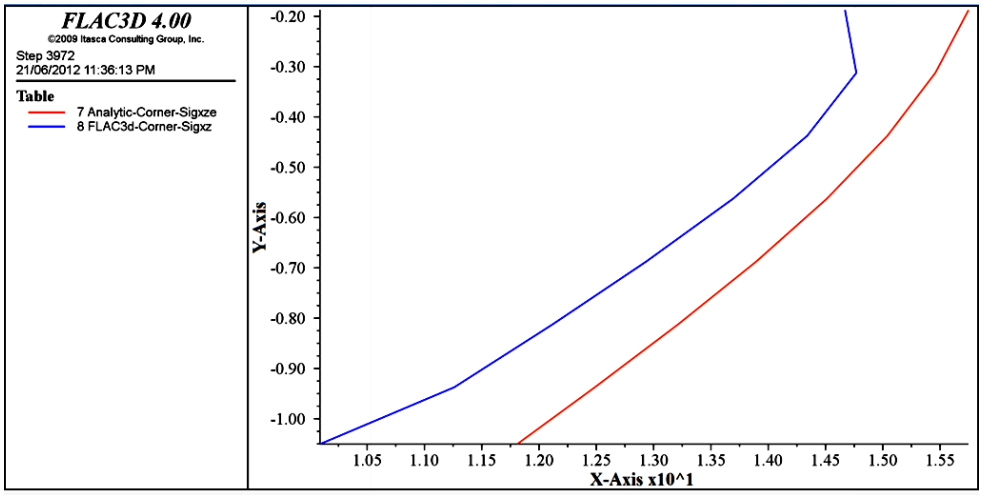


Figure 5.90 Uncoupled Flac3D and exact shear stress (σ_{xz}) solution under the corner of a uniform square load, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_{xz})*.

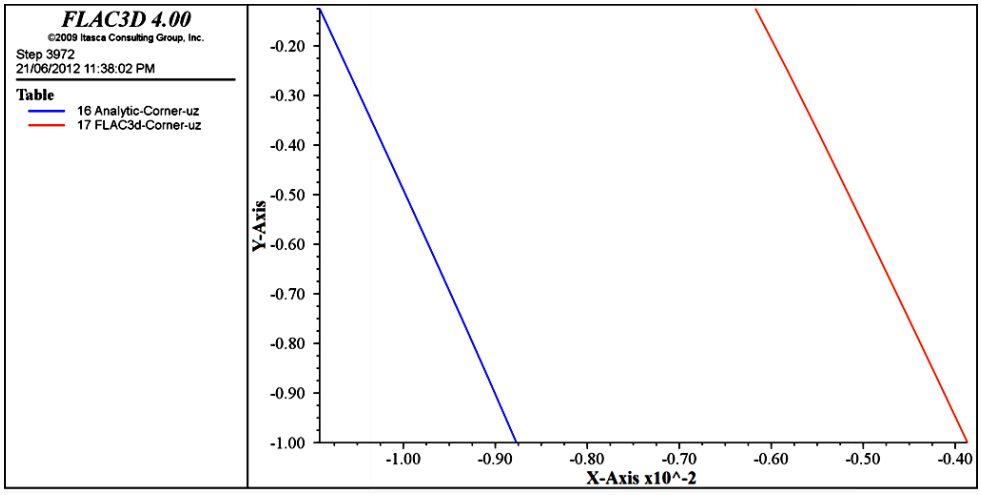


Figure 5.91 Uncoupled Flac3D and exact vertical displacement (u_z) solution under the corner of a uniform square load, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_z)*.

5.3.2.3 Coupled Flac^{3D} -BEM Solution (First Method/IDDM, ν = 0)

The first coupling Flac^{3D} -BEM method is applied on the same Flac^{3D} model, shown in Figure 5.84, with the same material properties (E = 10000 kN/m^2 and Poisson's ratio = 0). Generally, an accurate agreement (less than 1% RE) is achieved between the computed and the exact mechanical responses in both Flac^{3D} and BEM sub-domains. The vertical displacement u_z at point A and point B, shown in Figure 5.84, is corrected by almost 95% after applying this method compared to uncoupled Flac^{3D} solution (see Figures 5.92 and 5.93 and Table 5.20). The CPU in this solution competes with the uncoupled Flac^{3D} less accurate solution obtained with a very large size mesh and big ratio (R/a), see Table 5.20. Ten iterations are needed in the suggested coupling method when tolerance equals ϵ = 0.0025 and relaxation parameter equals ω = 0.4 to obtain the mentioned corrected solution.

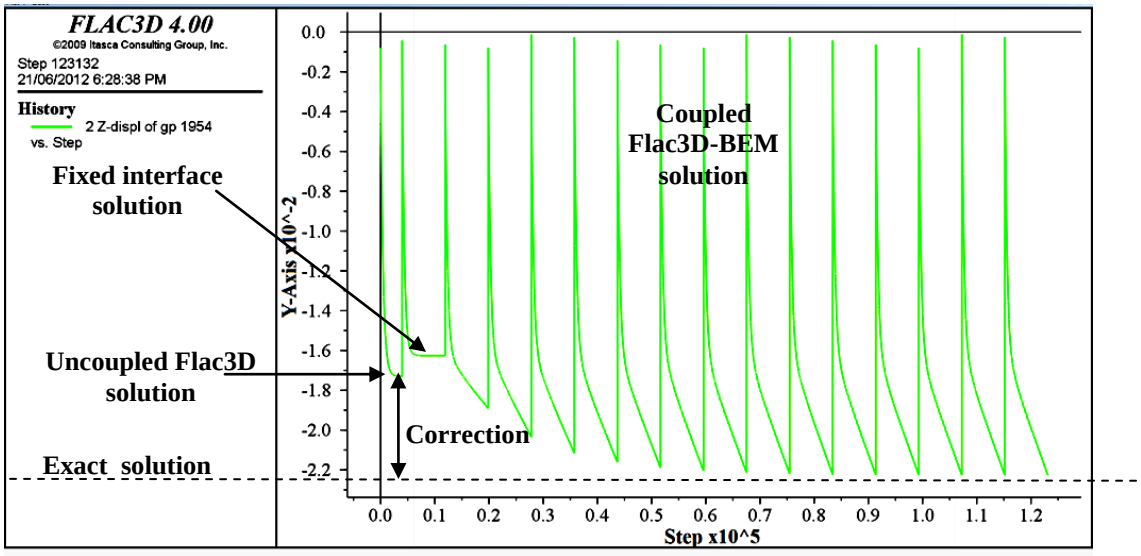


Figure 5.92 History of Flac3D and coupled (First Method/IDDM) vertical displacement at point A (the center of a uniform square load), ν = 0.0, *(Y-Axis ≡ u_z)*.

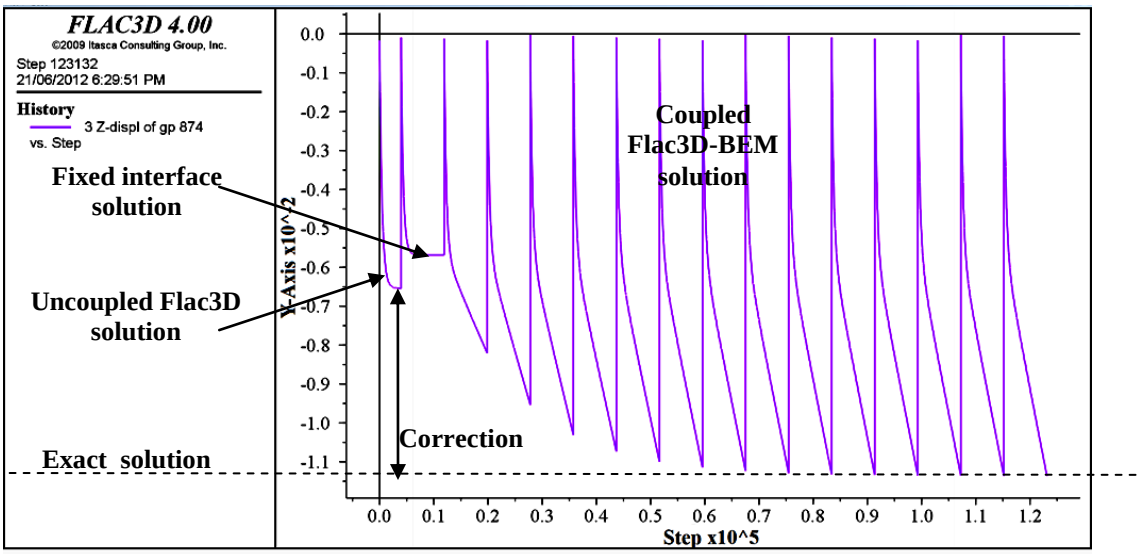


Figure 5.93 History of Flac3D and coupled (First Method/IDDM) vertical displacement at point B (the corner of a uniform square load), ν = 0.0, *(Y-Axis ≡ u_Z)*.

The vertical (σ_z) and horizontal stress (σ_x) under the load center in the Flac^{3D} sub-domain is corrected by this method significantly reducing the relative error for both stress components (see Figures 5.94 and 5.97). In particular, the horizontal stress solution in the Flac^{3D} sub-domain, which diverged from the exact solution with a big relative error (R E) in the uncoupled solution, is brought back to convergence proving again that the modification for σ_x made in equation 5.7 was accurate (see Figure 5.97 and Table 5.24). The stress continuity across the interface is fully developed for both stress components as shown in Figures 5.96 and 5.99, and the computed stress in the BEM sub-domain under the load center is almost 100 % exact (see Figures 5.95 and 5.98 and Table 5.24). Moreover, the stress (σ_z and σ_x) on the boundary, as a BEM post processing advantage, is computed with perfect agreement with the exact solution (see Table 5.24).

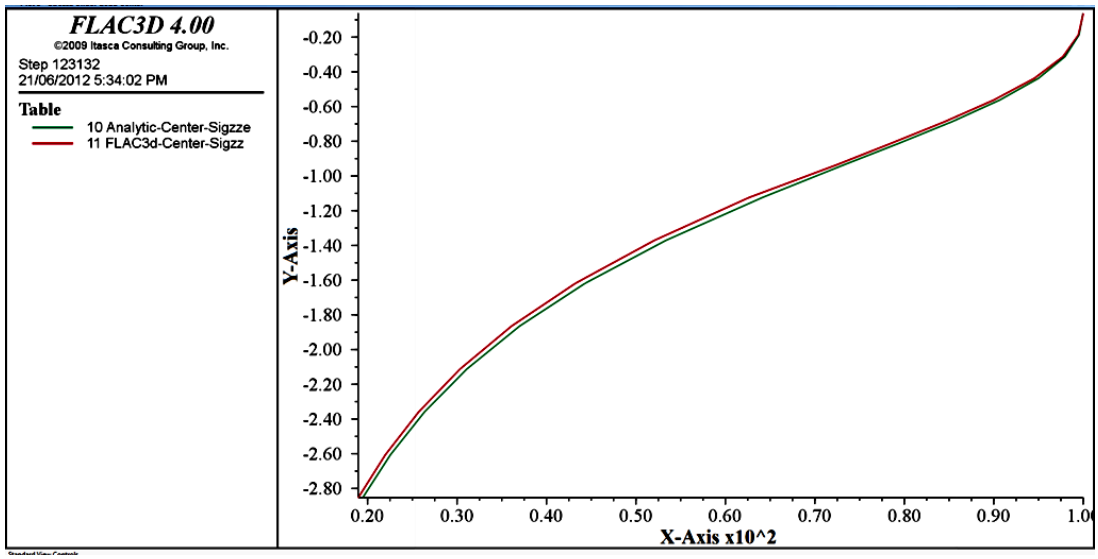


Figure 5.94 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solution under the center of a uniform square load in Flac3D sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

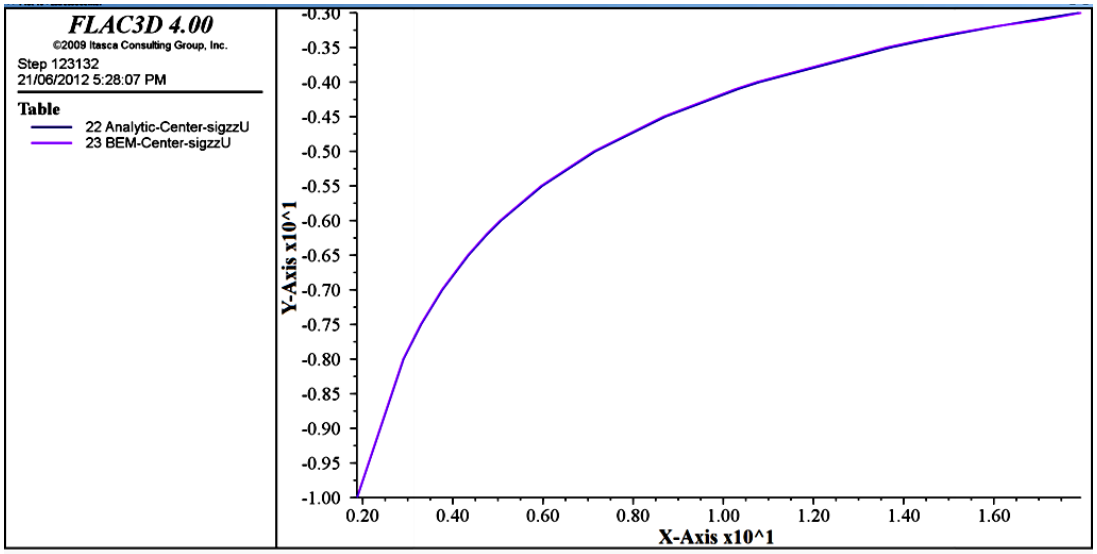


Figure 5.95 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solution under the center of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

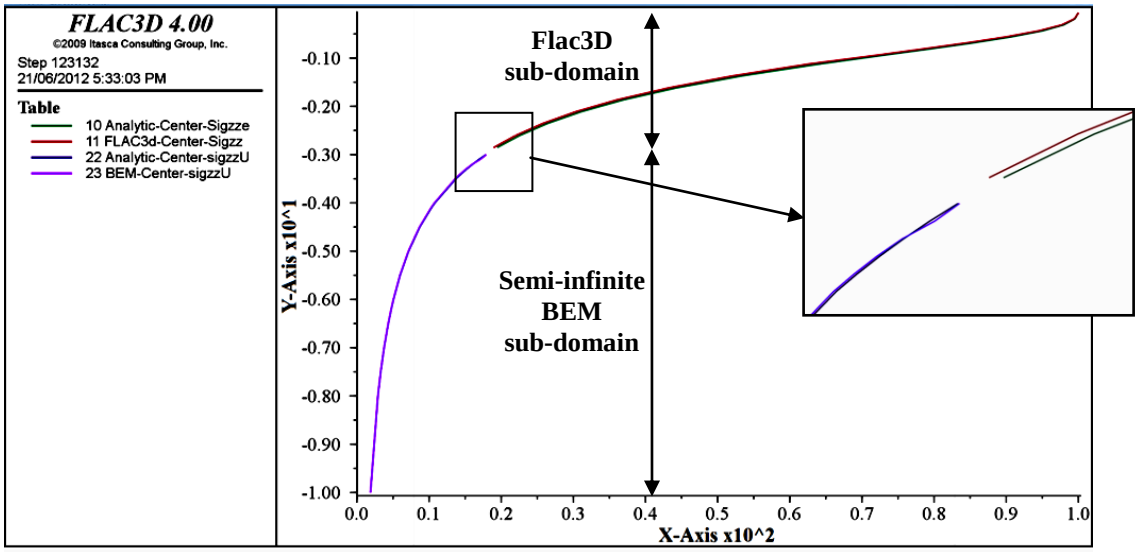


Figure 5.96 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.0,

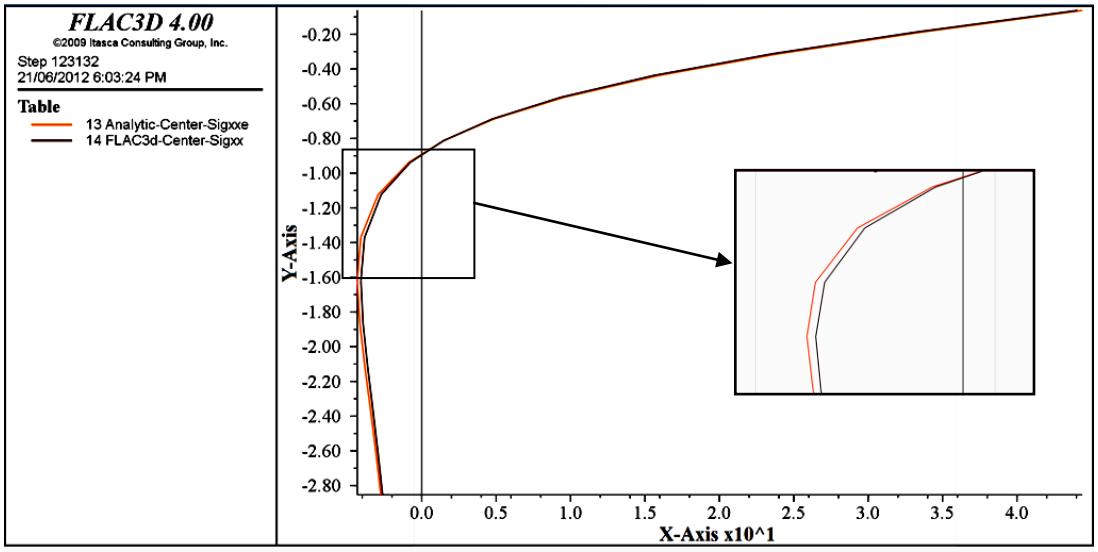


Figure 5.97 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solution under the center of a uniform square load in Flac3D sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

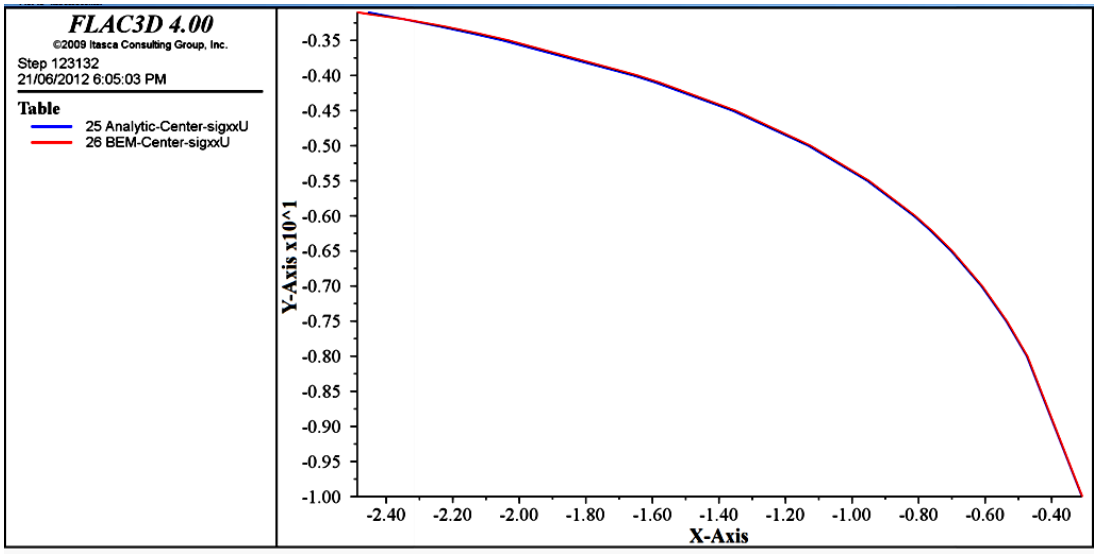


Figure 5.98 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solution under the center of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

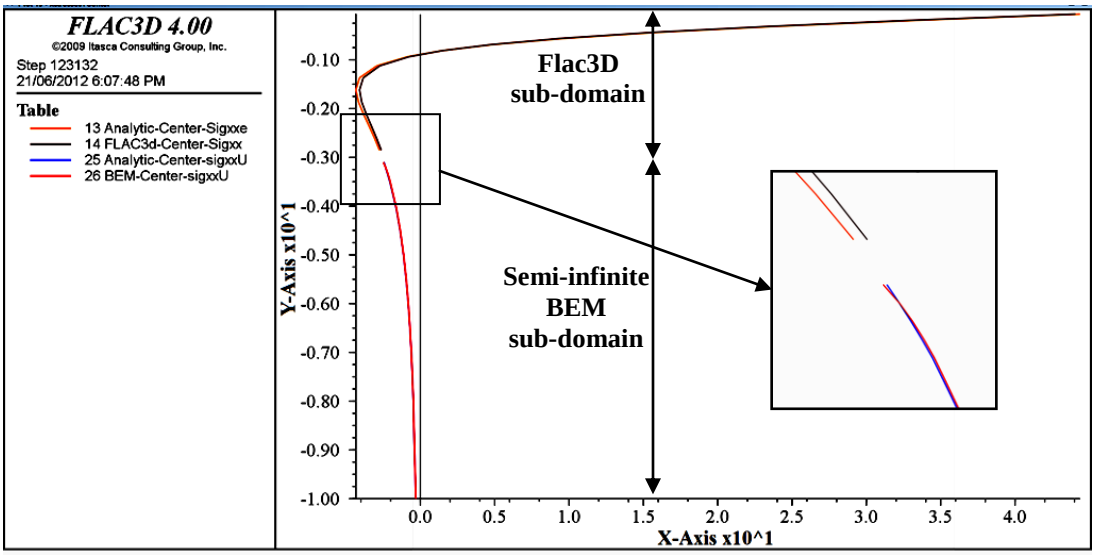


Figure 5.99 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solution under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

Table 5.24 Coupled (First Method/IDDM) and exact vertical (σ_z) and horizontal (σ_x) stress under the center of a loaded square area [kN/m2], ν = 0

	Depth [m]	σ_z Exact	σ_z Numerical	R.E%	σ_x Exact	σ_x Numerical	R.E%
	0.06	99.98	100.01	0.03	44.39	44.08	-0.70
	0.19	99.53	99.46	-0.07	33.60	33.21	-1.16
	0.31	97.97	97.73	-0.25	23.96	23.56	-1.65
	0.44	95.00	94.53	-0.49	15.91	15.58	-2.08
	0.56	90.67	90.00	-0.74	9.59	9.37	-2.31
	0.69	85.32	84.50	-0.96	4.86	4.76	-2.11
	0.81	79.36	78.49	-1.10	1.47	1.48	0.37
Flac^{3D}	0.94	73.17	72.35	-1.13	-0.86	-0.76	-10.77
sub-domain	1.12	64.16	62.67	-2.32	-2.93	-2.73	-6.92
	1.37	53.37	52.07	-2.42	-4.10	-3.84	-6.24
	1.62	44.33	43.25	-2.44	-4.33	-4.09	-5.61
	1.86	36.99	36.09	-2.43	-4.16	-3.95	-5.14
	2.11	31.10	30.34	-2.44	-3.82	-3.64	-4.84
	2.36	26.37	25.71	-2.48	-3.45	-3.29	-4.65
	2.61	22.55	21.98	-2.52	-3.08	-2.95	-4.51
	2.85	19.46	18.96	-2.58	-2.75	-2.63	-4.40
	3.00	17.89	17.94	0.27	-2.57	-2.60	1.33
	3.10	16.93	17.10	1.01	-2.45	-2.49	1.33
	3.20	16.03	15.99	-0.28	-2.34	-2.34	-0.04
	3.30	15.20	15.11	-0.61	-2.24	-2.23	-0.65
	3.40	14.43	14.34	-0.68	-2.14	-2.12	-0.84
	3.50	13.72	13.63	-0.68	-2.05	-2.03	-0.85
	4.00	10.81	10.74	-0.60	-1.66	-1.64	-0.77
	4.10	10.33	10.27	-0.58	-1.59	-1.58	-0.75
BEM	4.50	8.71	8.67	-0.53	-1.36	-1.35	-0.67
sub-domain	5.00	7.16	7.13	-0.47	-1.13	-1.12	-0.59
	5.50	5.98	5.96	-0.43	-0.95	-0.95	-0.52
	6.00	5.07	5.05	-0.40	-0.81	-0.81	-0.48
	6.20	4.76	4.74	-0.39	-0.77	-0.76	-0.46
	6.50	4.35	4.33	-0.38	-0.70	-0.70	-0.44
	7.00	3.77	3.76	-0.36	-0.61	-0.61	-0.41
	7.50	3.30	3.29	-0.34	-0.54	-0.53	-0.39
	8.00	2.91	2.90	-0.33	-0.47	-0.47	-0.37
	10.00	1.88	1.87	-0.31	-0.31	-0.31	-0.32

The vertical displacement u_z in Flac^{3D} sub-domain under the center and the corner of the load is also corrected by more than 98 % (see Figures 5.100 and 5.103). It is almost exact in the BEM sub-domain with an error less than 0.5 % (see Figures 5.101 and 5.104 and Tables 5.25 and 5.26). The displacement compatibility is clearly achieved across the interface by this method (see Figure 5.102).

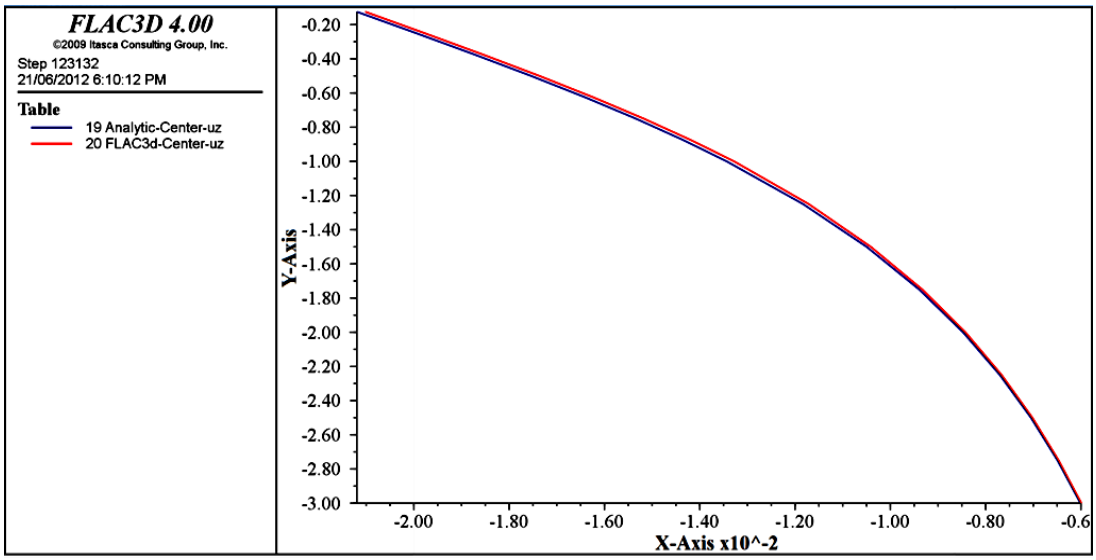


Figure 5.100 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solution under the center of a uniform square load in Flac3D sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z)*.

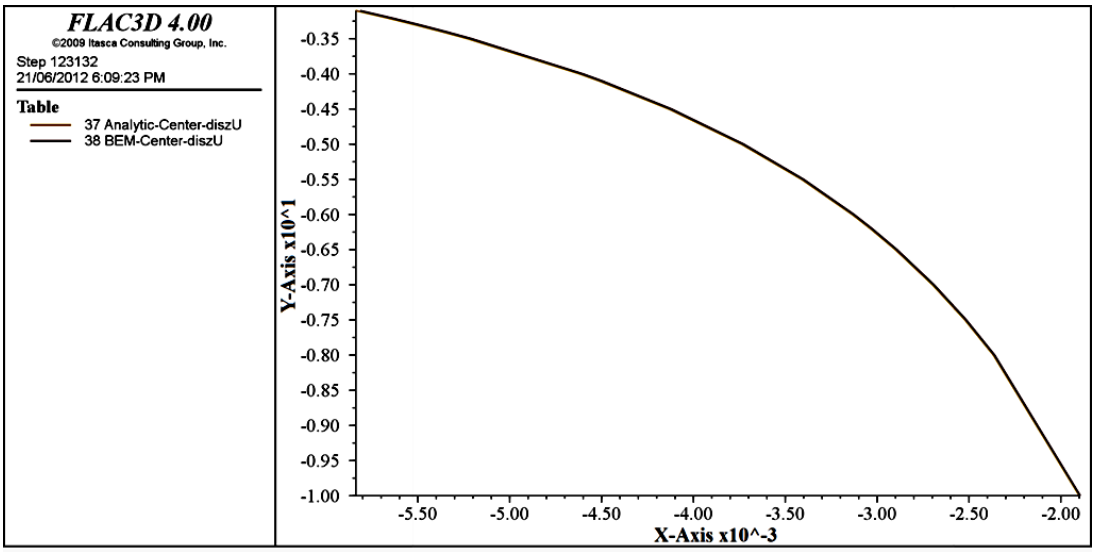


Figure 5.101 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solution under the center of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z)*

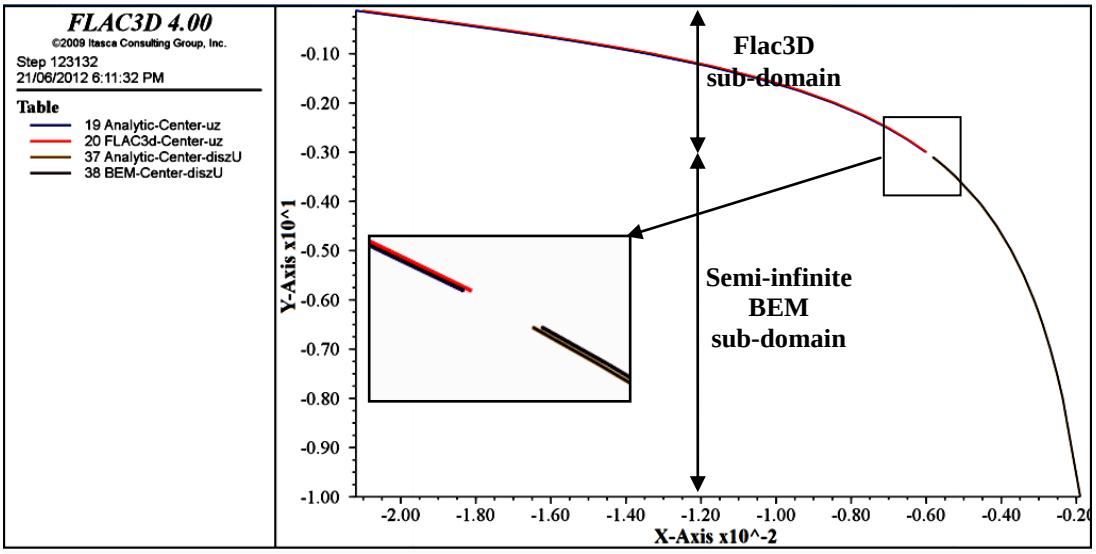


Figure 5.102 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains,

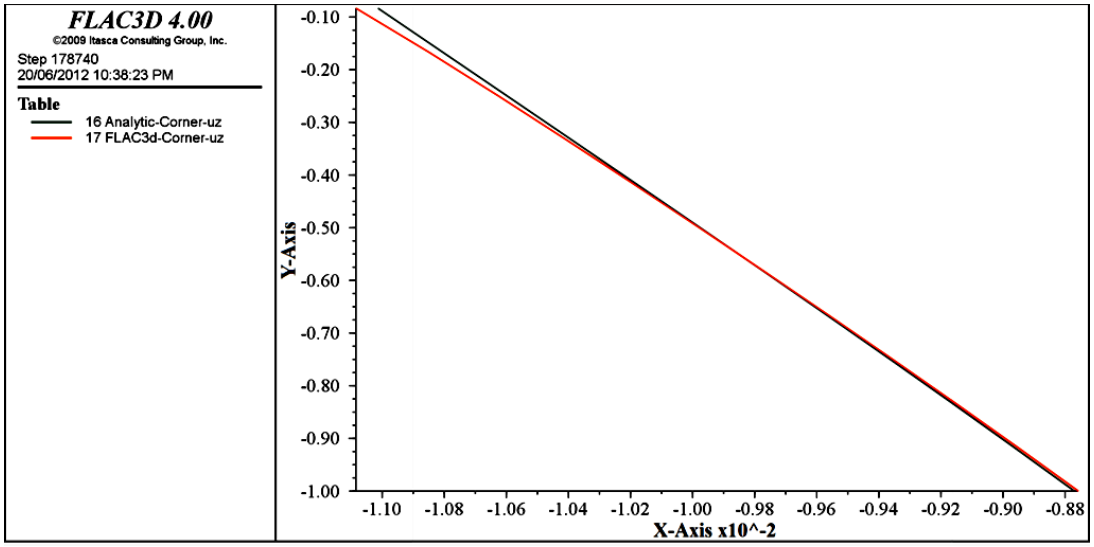


Figure 5.103 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solution under the corner of a uniform square load in Flac3D sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z)*

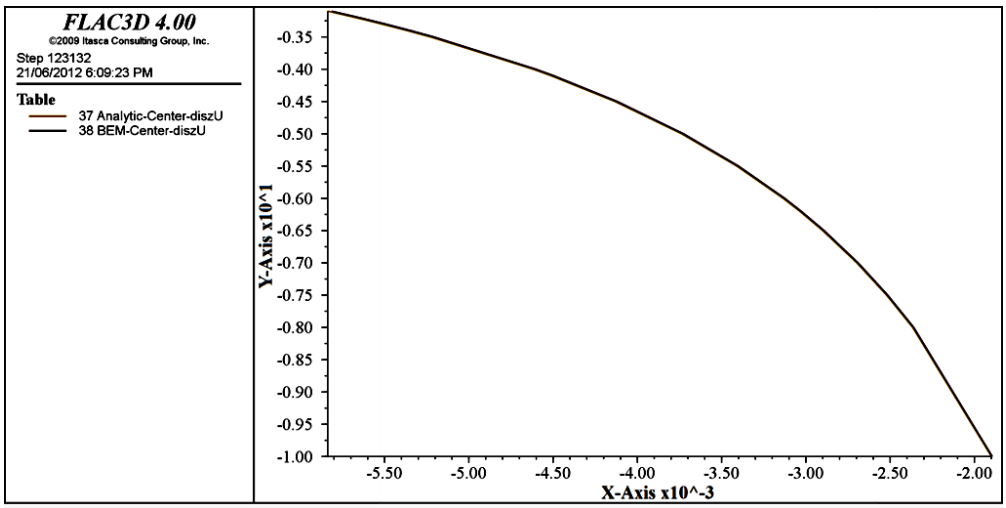


Figure 5.104 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solution under the corner of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z)*

Table 5.25 Coupled (First Method/IDDM) vertical displacement (u_z) under the corner of a loaded square area[m], ν = 0

	Depth [m]	u_z Exact	u_z Numerical	R.E%
Flac^{3D} sub-domain	0.13	-0.0109095	-0.0109963	0.80
	0.25	-0.0105972	-0.0106493	0.49
	0.38	-0.0102856	-0.0103161	0.30
	0.50	-0.0099755	-0.0099916	0.16
	0.63	-0.0096677	-0.0096739	0.06
	0.75	-0.0093635	-0.0093626	-0.01
	0.88	-0.009064	-0.0090581	-0.06
	1.00	-0.0087702	-0.0087614	-0.10
BEM sub-domain	2.8	-0.0055011	-0.0054819	-0.35
	3	-0.0052497	-0.0052313	-0.35
	3.1	-0.0051309	-0.0051130	-0.35
	3.2	-0.0050164	-0.0049991	-0.35
	3.3	-0.0049062	-0.0048893	-0.34
	3.4	-0.0047999	-0.0047836	-0.34
	3.5	-0.0046976	-0.0046817	-0.34
	4	-0.0042377	-0.0042238	-0.33
	4.1	-0.0041551	-0.0041416	-0.33
	4.5	-0.0038518	-0.0038395	-0.32
	5	-0.0035251	-0.0035142	-0.31
	5.5	-0.003246	-0.0032362	-0.30
	6	-0.0030056	-0.0029966	-0.30
	6.5	-0.0027966	-0.0027884	-0.29
	7	-0.0026137	-0.0026061	-0.29
	7.5	-0.0024523	-0.0024453	-0.29
	8	-0.0023092	-0.0023025	-0.29

Table 5.26 Coupled (First Method/IDDM) vertical displacement (u_z) under the center of a loaded square area[m], ν = 0

	Depth [m]	u_z Exact	u_z Numerical	R.E%
Flac^{3D} sub-domain	0.13	-0.0211944	-0.0210172	-0.84
	0.25	-0.0199509	-0.0197736	-0.89
	0.38	-0.0187271	-0.0185506	-0.94
	0.50	-0.0175403	-0.0173663	-0.99
	0.63	-0.0164075	-0.0162374	-1.04
	0.75	-0.0153413	-0.0151761	-1.08
	0.88	-0.0143494	-0.0141894	-1.12
	1.00	-0.0134347	-0.0132793	-1.16
	1.25	-0.0118311	-0.0117130	-1.00
	1.50	-0.0104993	-0.0104107	-0.84
	1.75	-0.0093951	-0.0093282	-0.71
	2.00	-0.0084754	-0.0084243	-0.60
	2.25	-0.0077036	-0.0076640	-0.51
	2.50	-0.0070502	-0.0070194	-0.44
	2.75	-0.0064921	-0.0064680	-0.37
	3.00	-0.0060111	-0.0059924	-0.31
BEM sub-domain	3.10	-0.0058371	-0.0058138	-0.40
	3.20	-0.0056723	-0.0056488	-0.42
	3.30	-0.0055162	-0.0054934	-0.41
	3.40	-0.0053681	-0.0053462	-0.41
	3.50	-0.0052273	-0.0052065	-0.40
	4.00	-0.0046183	-0.0046014	-0.37
	4.10	-0.0045126	-0.0044963	-0.36
	4.50	-0.004133	-0.0041188	-0.34
	5.00	-0.003738	-0.0037257	-0.33
	5.50	-0.0034107	-0.0033999	-0.32
	6.00	-0.0031353	-0.0031256	-0.31
	6.20	-0.003037	-0.0030277	-0.30
	6.50	-0.0029005	-0.0028917	-0.30
	7.00	-0.002698	-0.0026901	-0.30
	7.50	-0.0025218	-0.0025144	-0.29
	8.00	-0.0023669	-0.0023601	-0.29
	10.00	-0.0018994	-0.0018940	-0.28

Although the computed stress components σ_z, σ_x and σ_{xz} under the load corner in the BEM sub-domain are in perfect agreement with the exact solution with an error less than 1 % (see Figures 5.106, 5.108, and 5.110), the error was only reduced by 84-98 % for σ_x, 35- 84 % for $\sigma_{xz,}$ and

0.5-49 % for σ_z in the Flac^{3D} sub-domain (see Figures 5.107, 5.109, and 5.105). Conversely, the vertical displacement u_z under the load corner, as mentioned before, is corrected by more than 98 % in the Flac^{3D} sub-domain (see Figure 5.103). The maximum error in the stress under the load corner is observed to be at the surface (plane z = 0). This is attributed to the mesh configuration and how the Flac^{3D} program computes the stress.

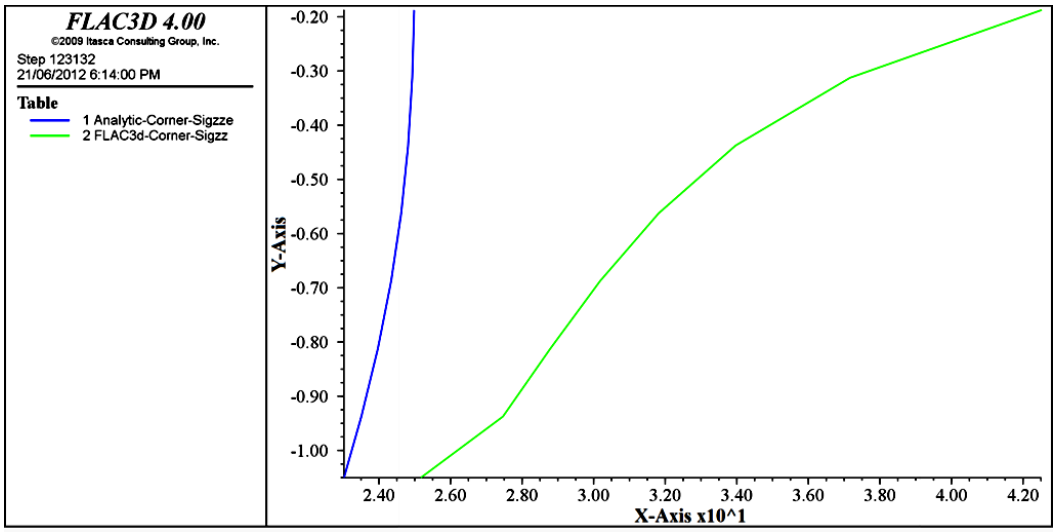


Figure 5.105 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solution under the corner of a uniform square load in Flac3D sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

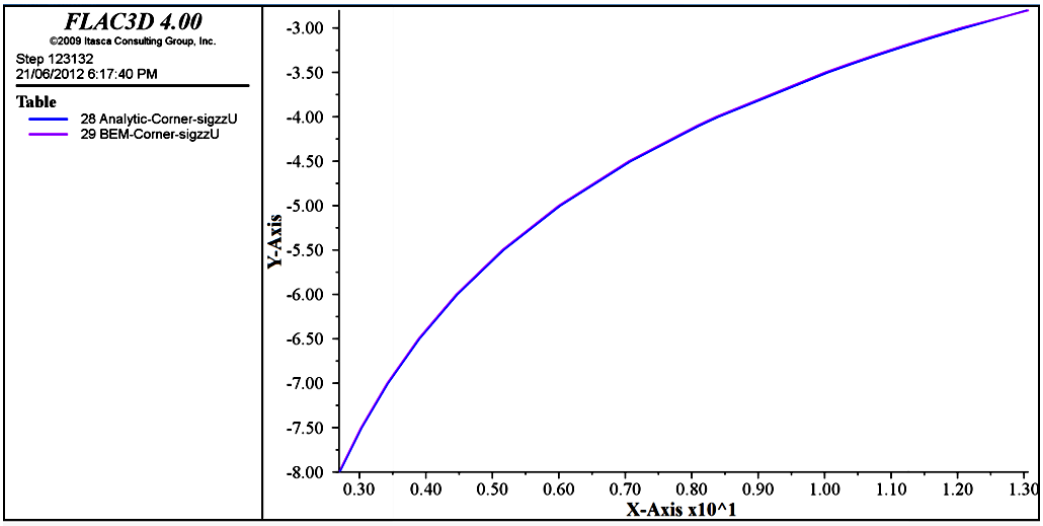


Figure 5.106 Coupled (First MethodIDDM) and exact vertical stress (σ_z) solution under the corner of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

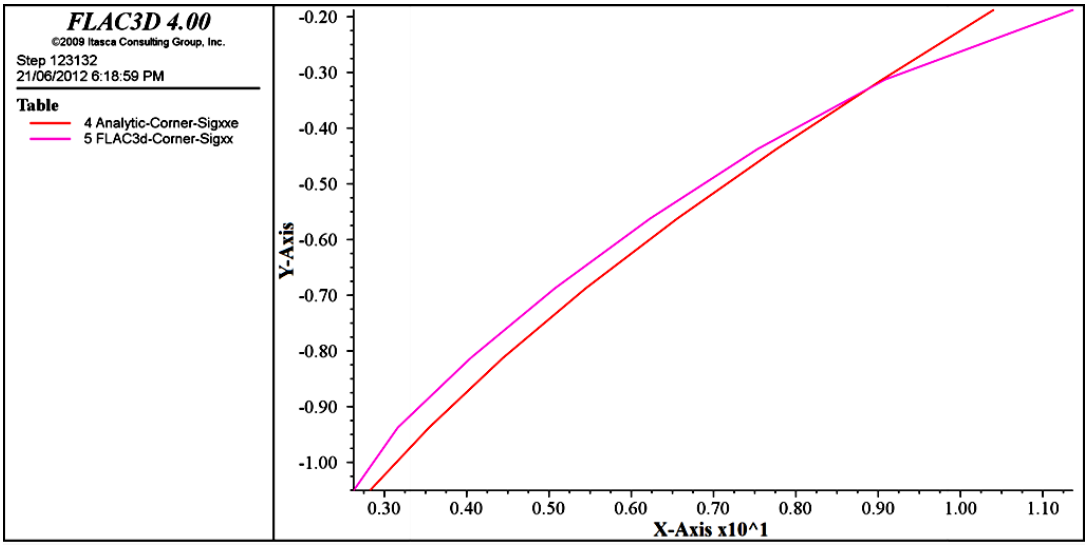


Figure 5.107 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solution under the corner of a uniform square load in Flac3D sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

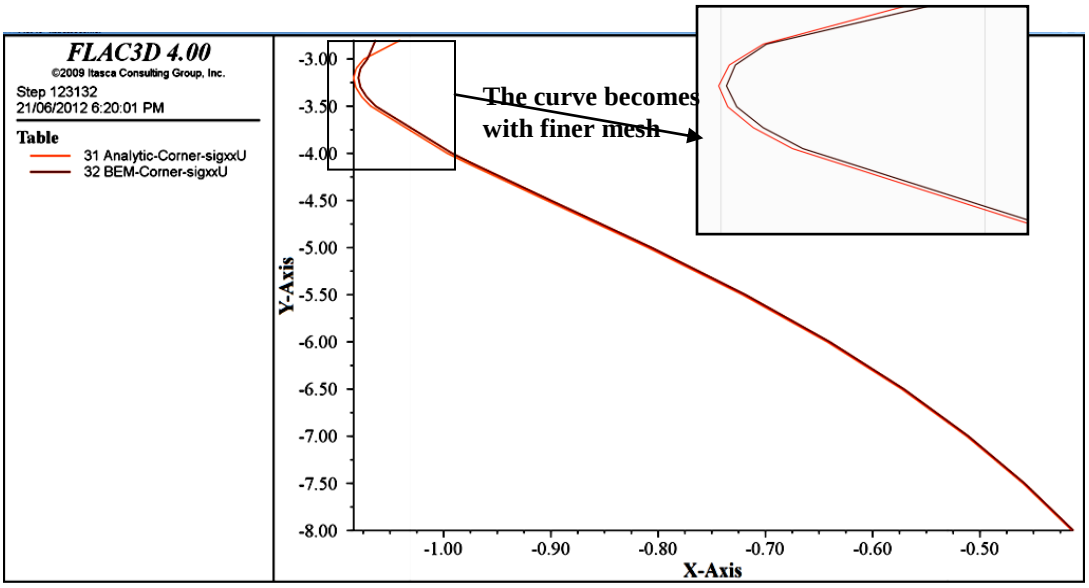


Figure 5.108 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solution under the corner of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

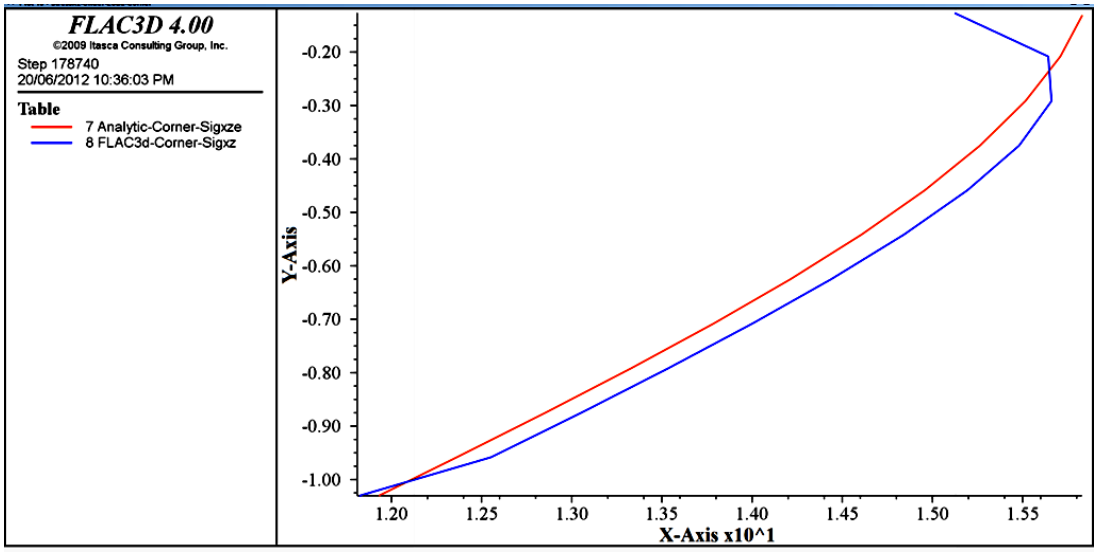


Figure 5.109 Coupled (First Method/IDDM) and exact shear stress (σ_{xz}) solution under the corner of a uniform square load in Flac3D sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_{xz})*.

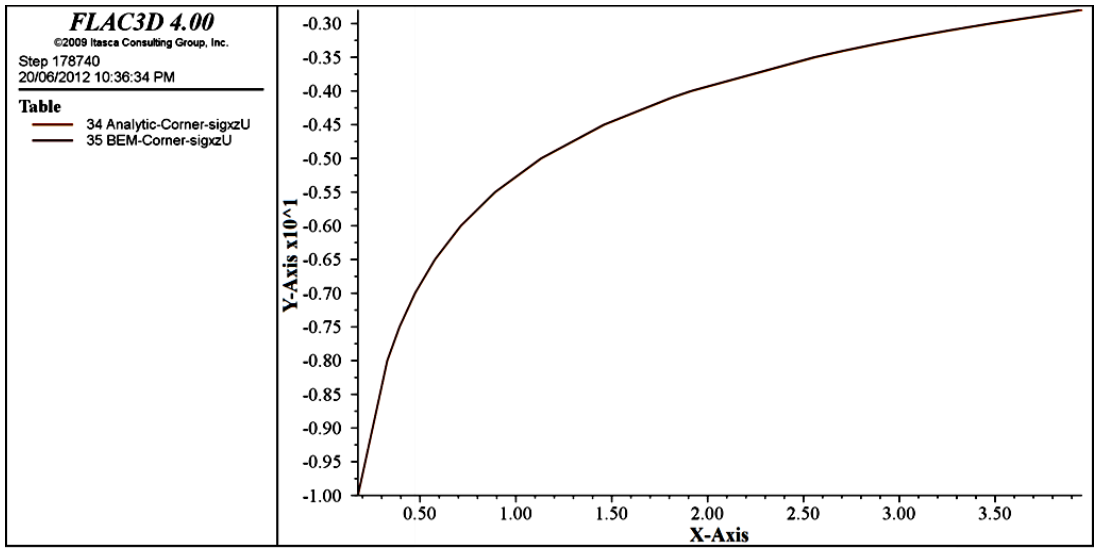


Figure 5.110 Coupled (First Method/IDDM) and exact shear stress (σ_{xz}) solution under the corner of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_{xz})*.

Iterations convergence test: The test conducted on the same Flac^{3D} model, seen in Figure 5.84, proved that the solution of the first coupling method is independent from the number of renewal iterations. This is inferred from the observed stability of the vertical displacement values u_z at points A, B and a number of points in both Flac^{3D} and BEM sub-domains, as shown in Figures 5.112-5.114, the vertical stress σ_z and the horizontal stress σ_x at a number of zones in both sub-domains, as shown in Figures 5.115-5.118 (see Appendix c, pp.323-324). The positions of the test points and zones can be seen in Table 5.27 and Figure 5.111. However, the stress tends to converge faster than the displacement as observed in the above named figures. The minimum number of iterations (NIT) needed to reduce the initial error by a factor $\eta = 10^{-2}$ equals 10 and the average reduction factor per iteration (E. R. F.) equals 0.625.

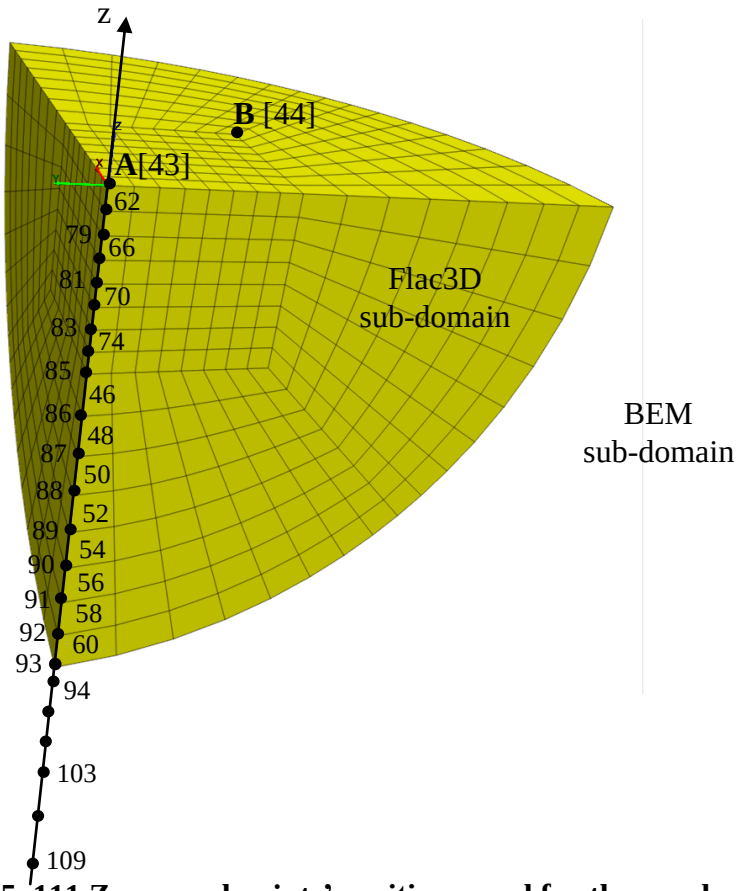


Figure 5. 111 Zones and points' positions used for the number of iterations convergence test

Table 5.27 Points' positions under the center of a uniform square load used for the iteration convergence test

	Depth	Zone No. for σ_z test	Zone No. for σ_x test
Flac^{3D} sub-domain	-0.06	62	63
	-0.19	64	65
	-0.31	66	67
	-0.44	68	69
	-0.56	70	71
	-0.69	72	73
	-0.81	74	75
	-0.94	76	77
	-1.12	46	47
	-1.37	48	49
	-1.62	50	51
	-1.86	52	53
	-2.11	54	55
	-2.36	56	57
	-2.61	58	59
	-2.85	60	61
BEM sub-domain	-3.10	94	95
	-3.20	97	98
	-3.30	100	101
	-3.40	103	104
	-3.50	106	107
	-4.00	109	110
	-4.10	112	113
	-4.50	115	116
	-5.00	118	119
	-5.50	121	122
	-6.00	124	125
	-6.20	127	128
	-6.50	130	131
	-7.00	133	134
	-7.50	136	137
	-8.00	139	140
	-10.00	142	143

	Depth	Point No. for u_z test
Flac^{3D} sub-domain	-0.13	78
	-0.25	79
	-0.38	80
	-0.50	81
	-0.63	82
	-0.75	83
	-0.88	84
	-1.00	85
	-1.25	86
	-1.50	87
	-1.75	88
	-2.00	89
	-2.25	90
	-2.50	91
	-2.75	92
	-3.00	93
BEM sub-domain	-3.10	96
	-3.20	99
	-3.30	102
	-3.40	105
	-3.50	108
	-4.00	111
	-4.10	114
	-4.50	117
	-5.00	120
	-5.50	123
	-6.00	126
	-6.20	129
	-6.50	132
	-7.00	135
	-7.50	138
	-8.00	141
	-10.00	144

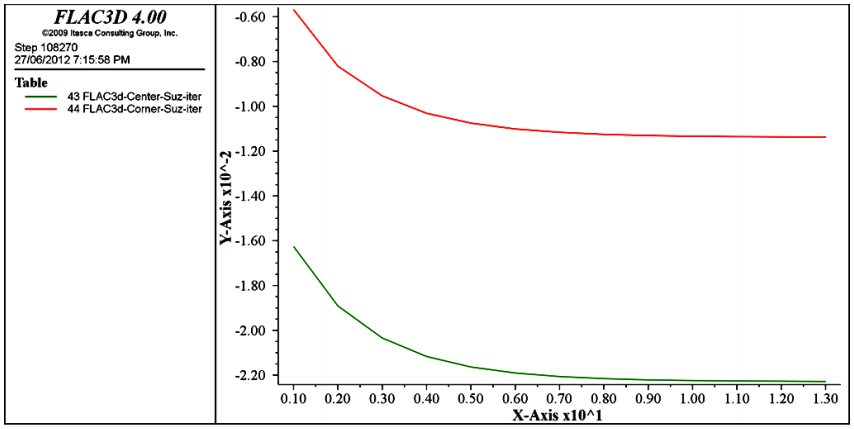


Figure 5.112 Convergence of the vertical displacement (u_z) at point A [43] and point B [44] on the surface over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ u_Z).*

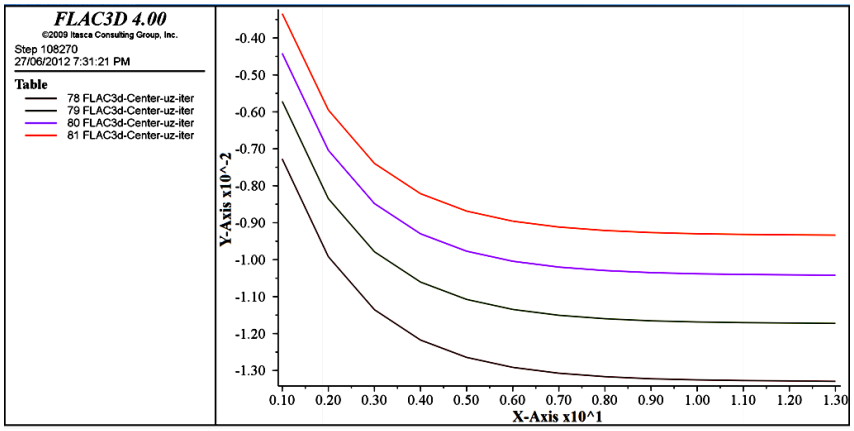


Figure 5.113 Convergence of vertical displacement (u_z) at chosen points in the Flac3D sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ u_z).*

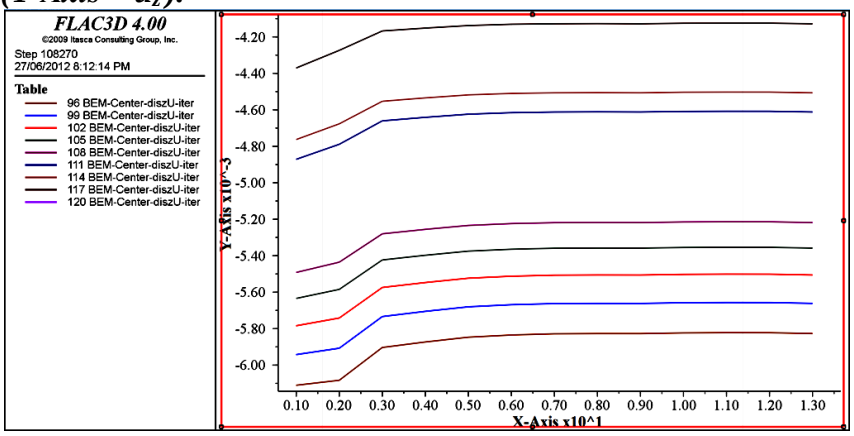


Figure 5.114 Convergence of vertical displacement (u_z) at chosen points in the BEM sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ u_z).*

5.3.2.4 Coupled Flac^{3D} -BEM Solution (First Method/IDDM), ν = 0.3)

The square load on a semi-infinite domain problem is solved with different property materials (E = 10000 kN/m^2 and ν= 0.3) to verify if the first coupling BEM-Flac^{3D} method is accurately approximating the effect of Poisson's ratio (υ) on the loaded semi-infinite domain's mechanical response presented by Giroud [127] and discussed before in section 5.3.2.1. The agreement between the computed mechanical response by this method and the exact one in both Flac^{3D} and BEM sub-domains in the case with a non-zero-Poisson's- ratio value, in general, is not different from that in the case with a zero-Poisson's- ratio value analysed numerically before. The vertical displacement u_z at point A and point B is corrected by almost 95 % after applying the IDD method compared to the uncoupled Flac^{3D} solution (see Figures 5.119 and 5.120; and Table 5.28). Ten iterations are needed in the suggested coupling method when tolerance equals $\epsilon = 0.0025$ and relaxation parameter equals $\omega = 0.4$ to obtain the mentioned corrected solution.

Table 5.28 Vertical displacement u_z under a loaded square area obtained by two methods, ν = 0.3

Uncoupled Flac^{3D} Solution				
Point	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**Number of zones**
A [Center]	0.020424035	0.01616747	-20.84	2048
B [Corner]	0.010212017	0.00624155	-38.88	
Coupled BEM-Flac^{3D} Solution [IDD Method]				
Point	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**Number of zones**
A [Center]	0.02042403	0.02029550	-0.63	2048
B [Corner]	0.01021202	0.01033982	1.25	

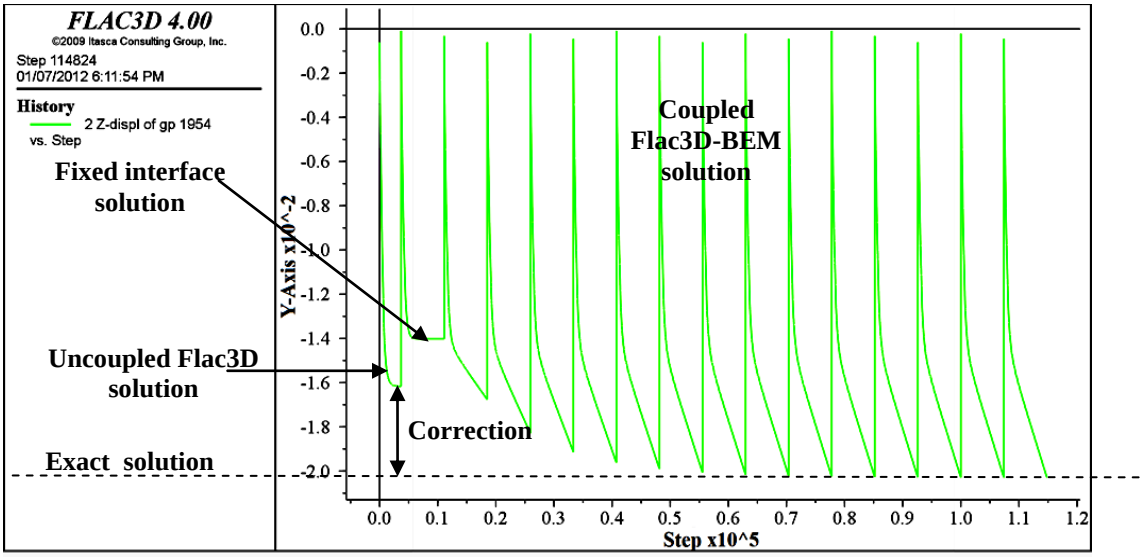


Figure 5.119 History of Flac3D and coupled (First Method/IDDM) vertical displacement u_Z at point A (the center of a uniform square load), ν = 0.3, *(Y-Axis ≡ u_Z).*

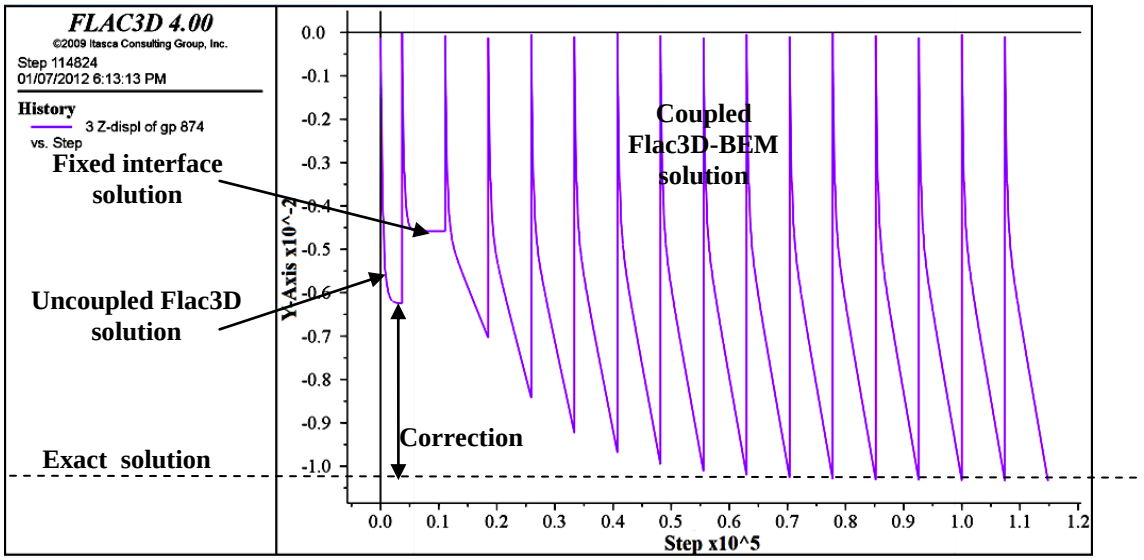


Figure 5.120 History of Flac3D and coupled (First Method/IDDM) vertical displacement u_Z at point B (the corner of a uniform square load), ν = 0.3, *(Y-Axis ≡ u_Z).*

Because Poisson's ratio has a minor effect over σ_z values, as mentioned before, the vertical stress σ_z curve, plotted according to equations (5.9) and named Analaytic-Center-Sigzze-ny, and σ_z curve, plotted according to equations (5.6) and named Analaytic-Center-Sigzze are

identical (see Figure 5.121). After applying the IDDM (first coupling) method, The vertical and horizontal stress (σ_z and σ_x, respectively) under the load center in Flac^{3D} sub-domain matches, the exact solution of equations (5.9) (less than 1% RE; see Figures 5.121 and 5.124; and Table 5.30 in Appendix c, p.326). Moreover, both computed stress components in the BEM sub-domain under the load center are almost 100 % exact (see Figures 5.122 and 5.125; and Table 5.29 in Appendix c, p.325). The plotted stress (σ_x) that curves according to equations (5.9) is named Analaytic-Center-Sigxxe-ny. The stress continuity across the interface is fully developed for both stress components as shown in Figures 5.123 and 5.126, and the stress (σ_z and σ_x) on the boundary is computed with perfect agreement with the exact solution (see Table 5.29 in Appendix c p.325).

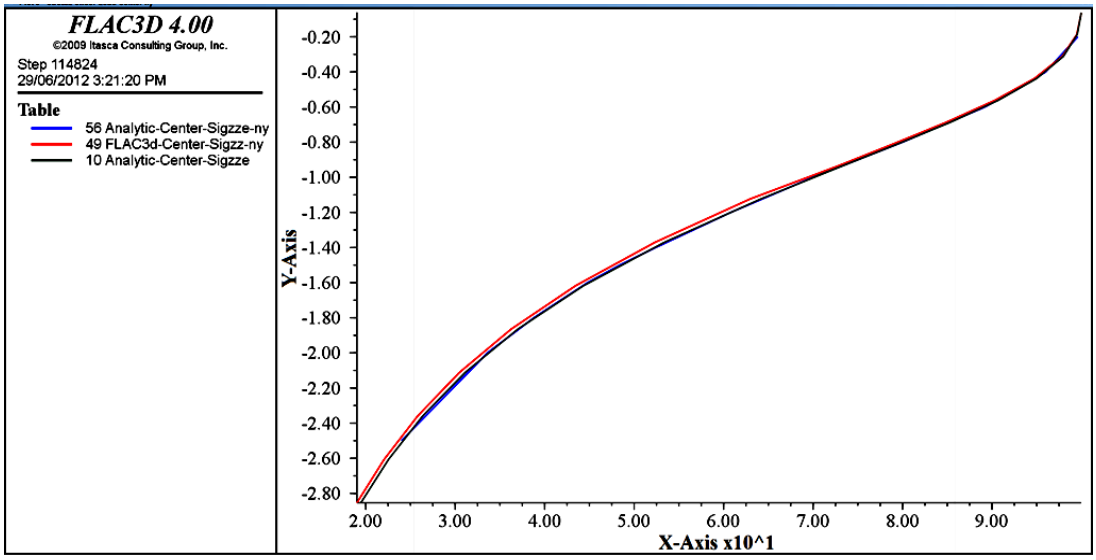


Figure 5.121 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solutions under the center of a uniform square load in Flac3D sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z)*

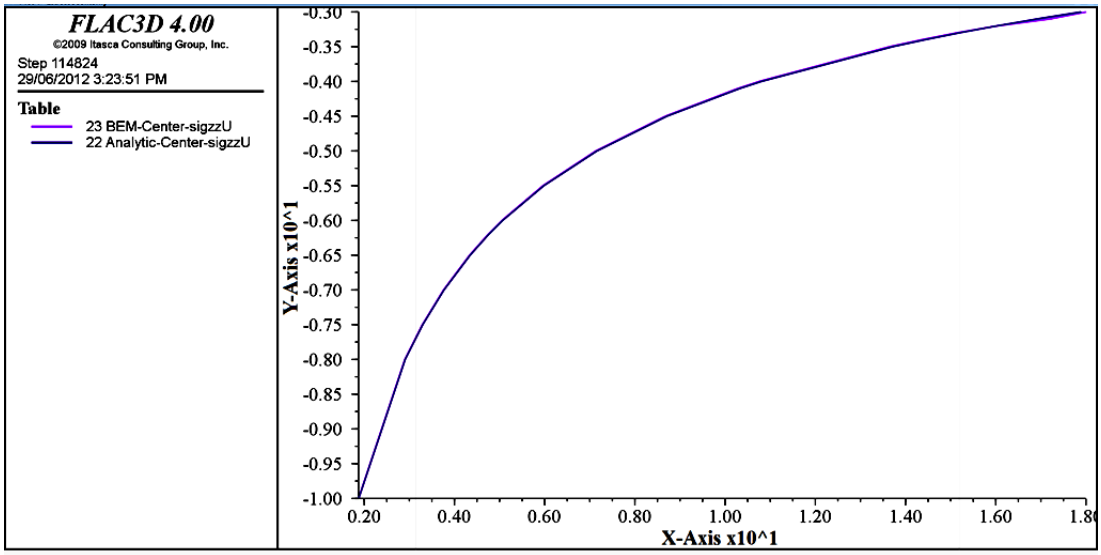


Figure 5.122 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solutions under the center of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

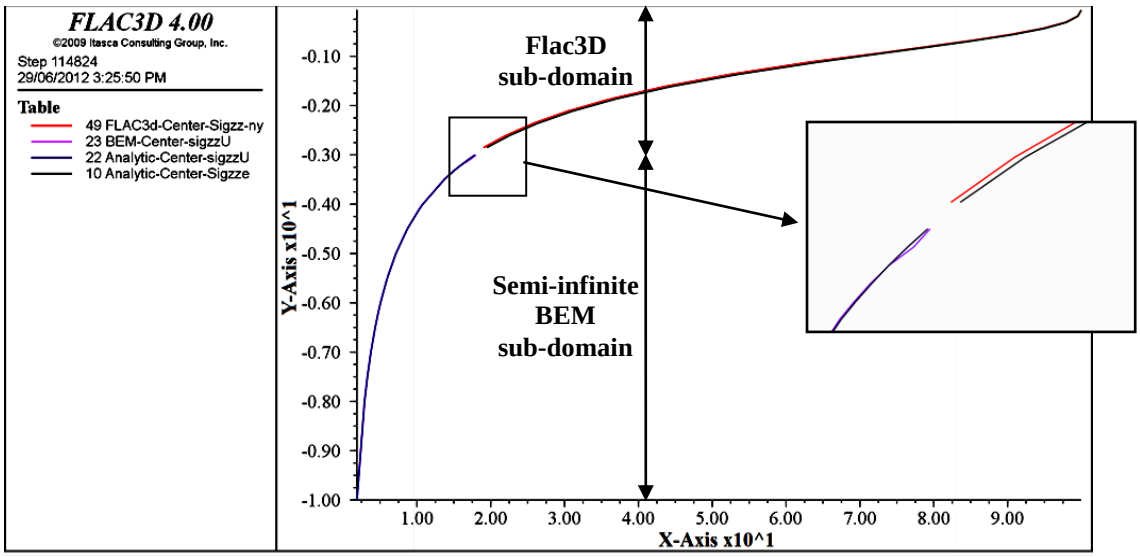


Figure 5.123 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z).*

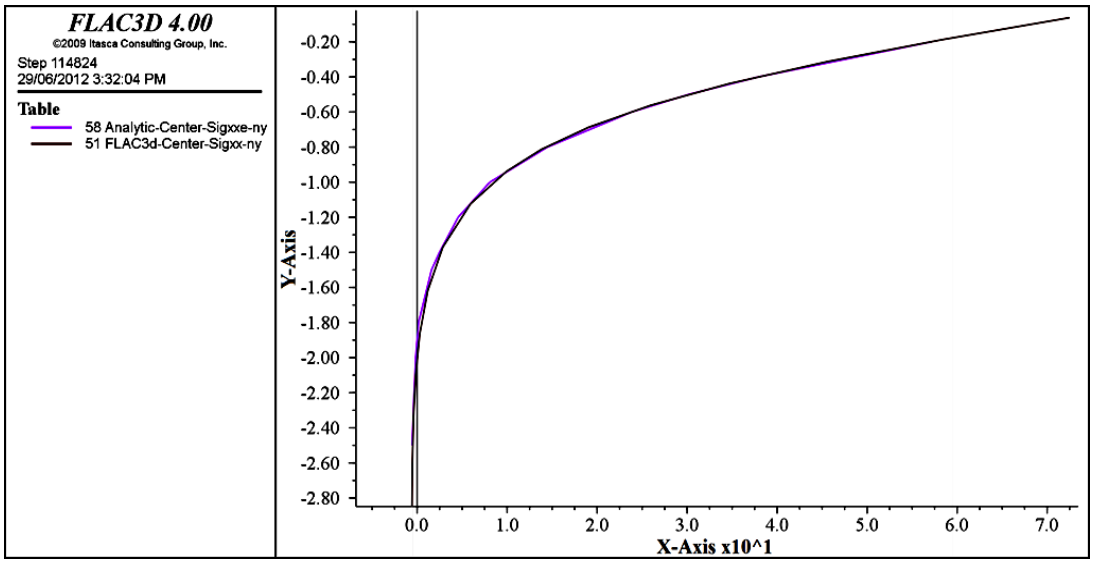


Figure 5.124 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solutions under the center of a uniform square load in Flac3D sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

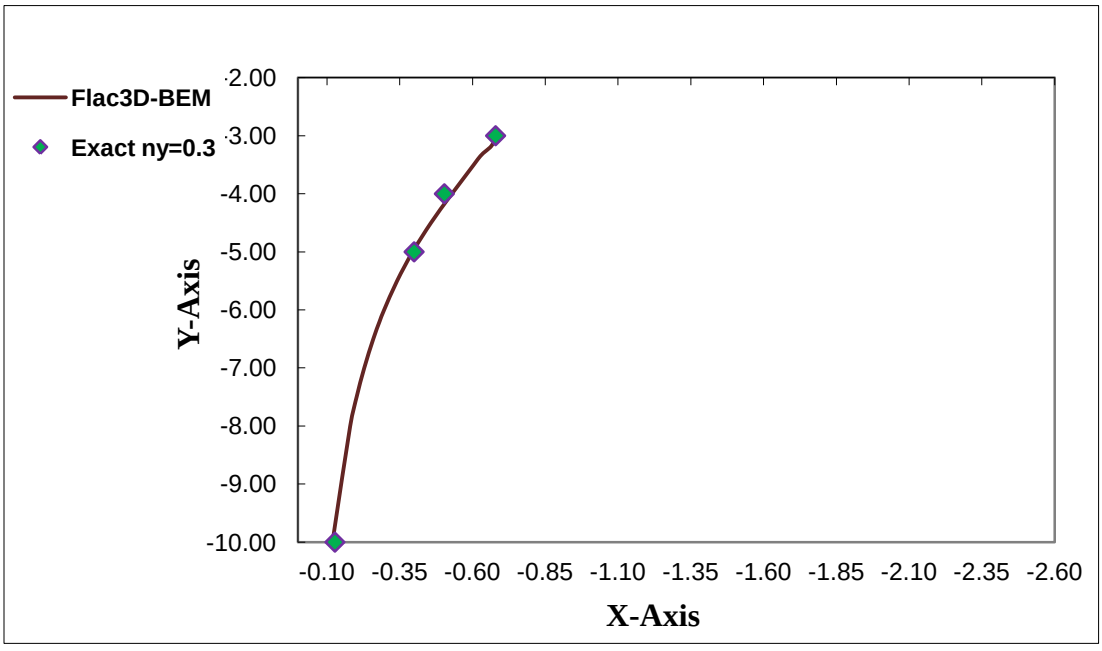


Figure 5.125 Coupled (FirstMethod/IDDM) and exact horizontal stress (σ_x) solutions under the center of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

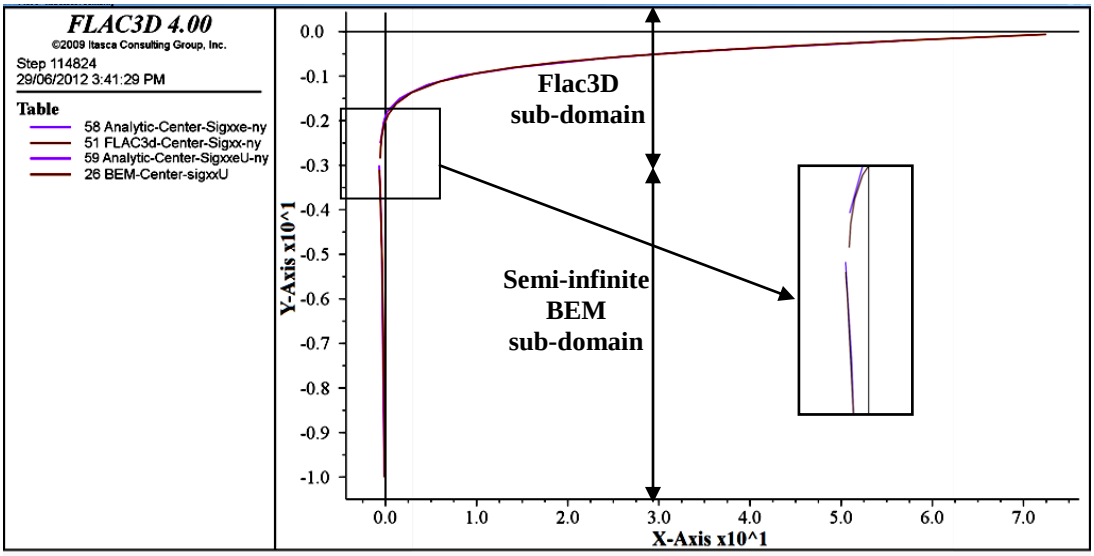


Figure 5.126 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solutions under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

The vertical displacement u_z in the Flac^{3D} sub-domain under the center and the corner of the load is corrected, similar to the case with a zero-Poisson's- ratio value, by more than 98 % compared to the uncoupled solution (see Figures 5.127 and 5.130; and Tables 5.31 and 5.32 in Appendix c, pp.327-328).

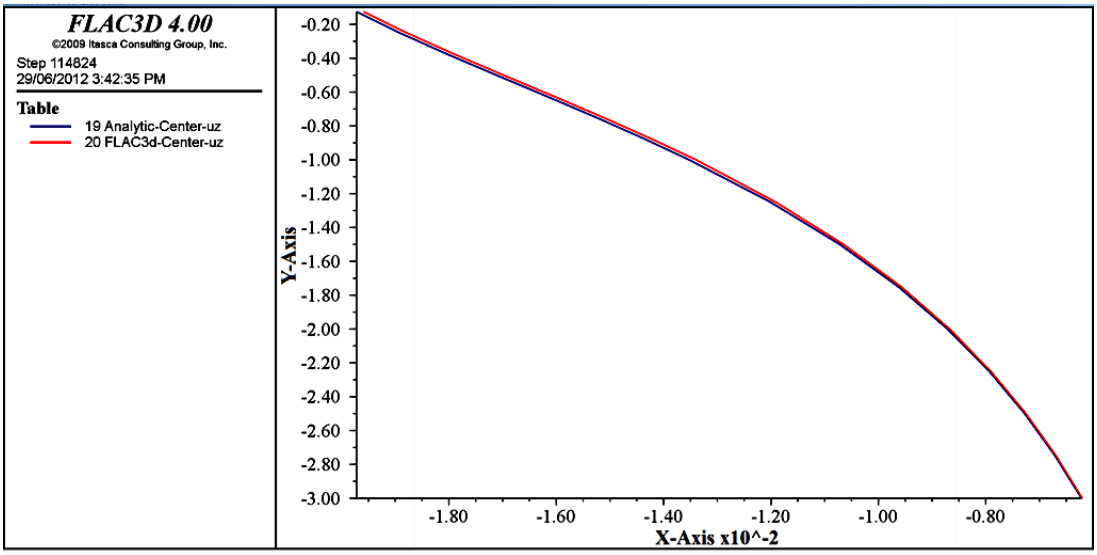


Figure 5.127 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solutions under the center of a uniform square load in Flac3D sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z).*

The coupled solution is almost exact in the BEM sub-domain (see Figures 5.128 and 5.131). Furthermore, the displacement compatibility is obviously achieved across the interface by this method (see Figure 5.129).

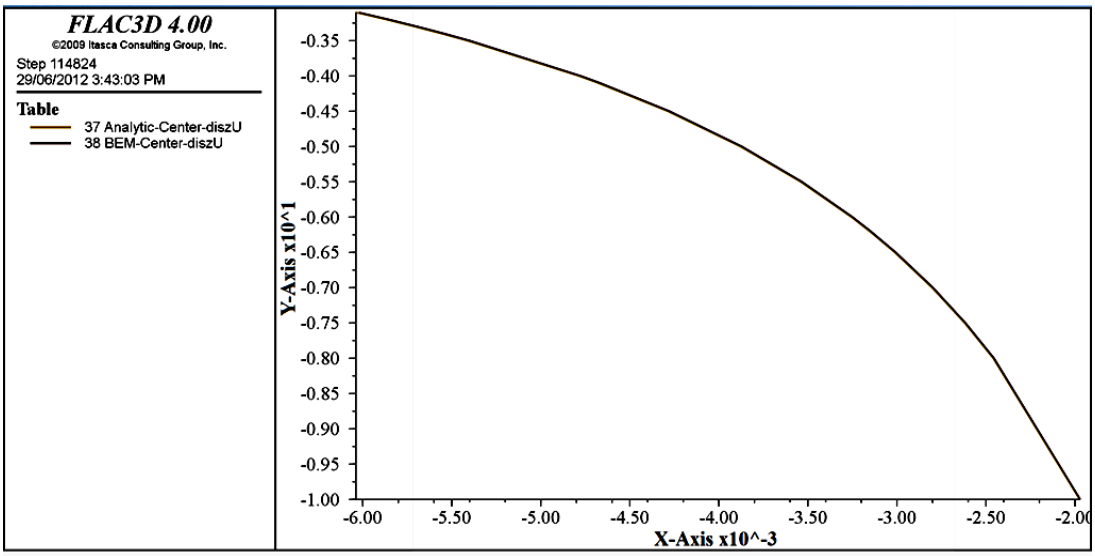


Figure 5.128 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solutions under the center of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z)*

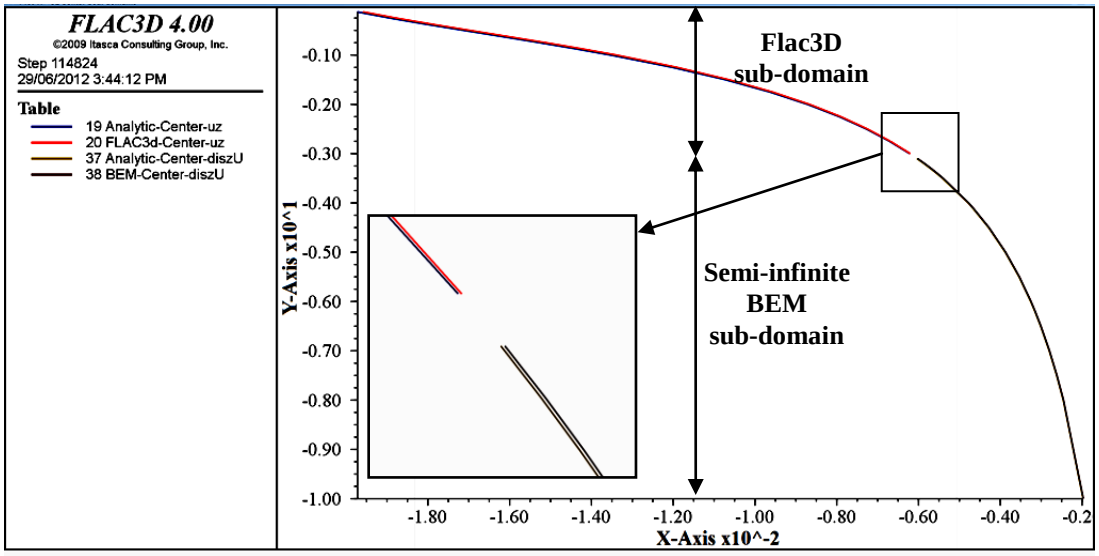


Figure 5.129 Coupled (First Mehtod/IDDM) and exact vertical displacement (u_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains,

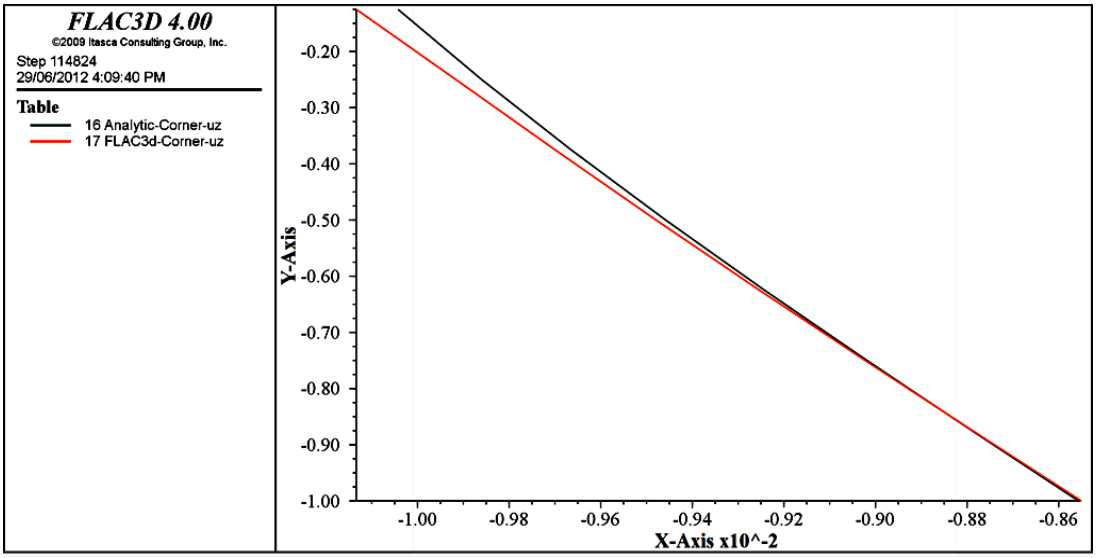


Figure 5.130 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solution under the corner of a uniform square load in Flac3D sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z)*

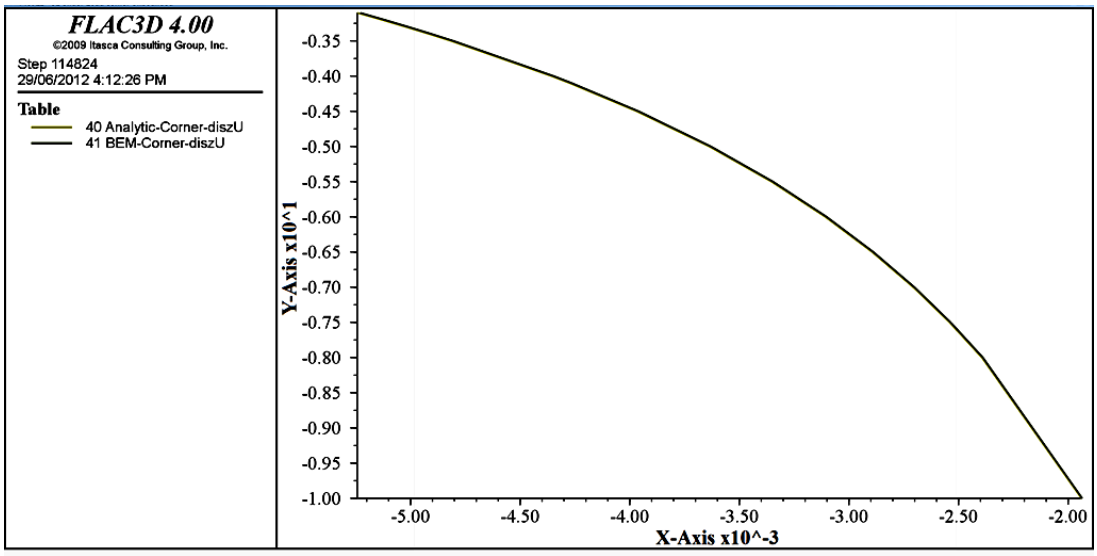


Figure 5.131 Coupled (First Method/IDDM) and exact vertical displacement (u_z) solution under the corner of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z)*

The vertical stress σ_z under the load corner in the BEM sub-domain is in perfect agreement with the exact solution (see Figure 5.133). The analytical solution of equation 5.9 is given the name Analytic-Corner-sigxxU-ny and the numerical one is named BEM-Corner-sigxxU. The last two solutions agree perfectly with each other (see Figure 5.135). Although Giroud [127] did not present an analytical solution for the shear stress $\sigma_{xz,}$ the analytical solution of equation 5.6 and the coupled one agree completely representing another indication that Poisson's ratio has minor effect in the vertical or the load direction (see Figure 5.137). The same observation about the stress components in the Flac^{3D} sub-domain recorded before in the zero-Poisson's- ratio case applies to this case too (see Figures 5.132, 5.134, and 5.136).

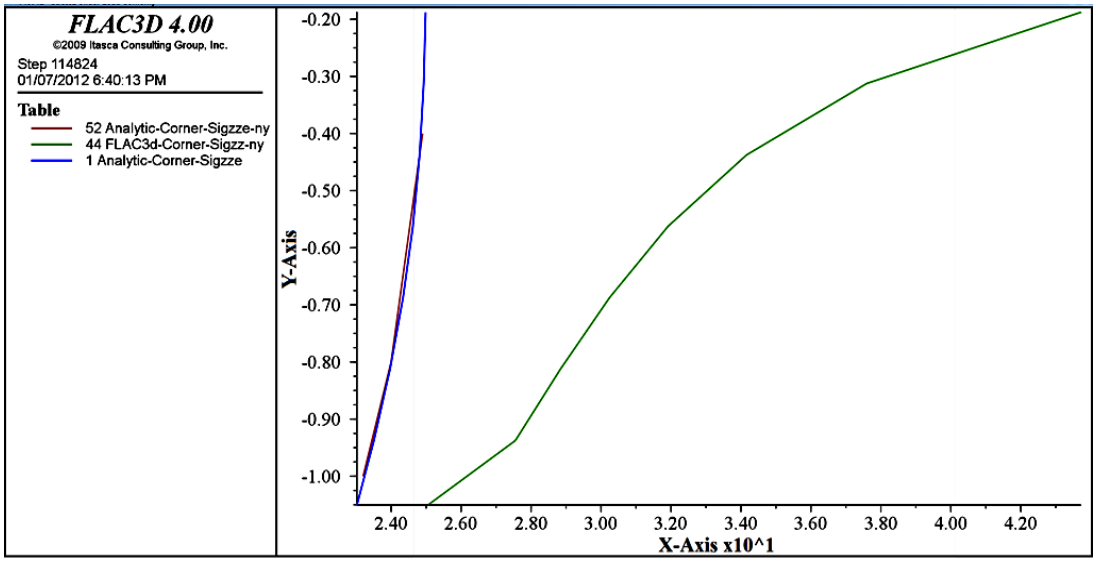


Figure 5.132Coupled (FirstMethod/IDDM) and exact vertical stress (σ_z) solutions under the corner of a uniform square load in Flac3D sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_z)*

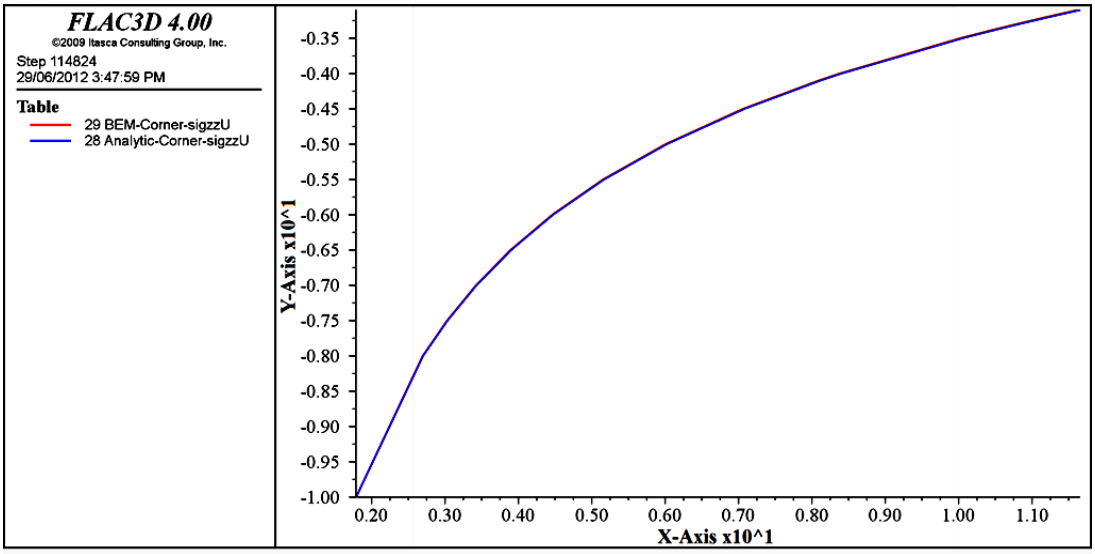


Figure 5.133 Coupled (First Method/IDDM) and exact vertical stress (σ_z) solutions under the corner of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z)*

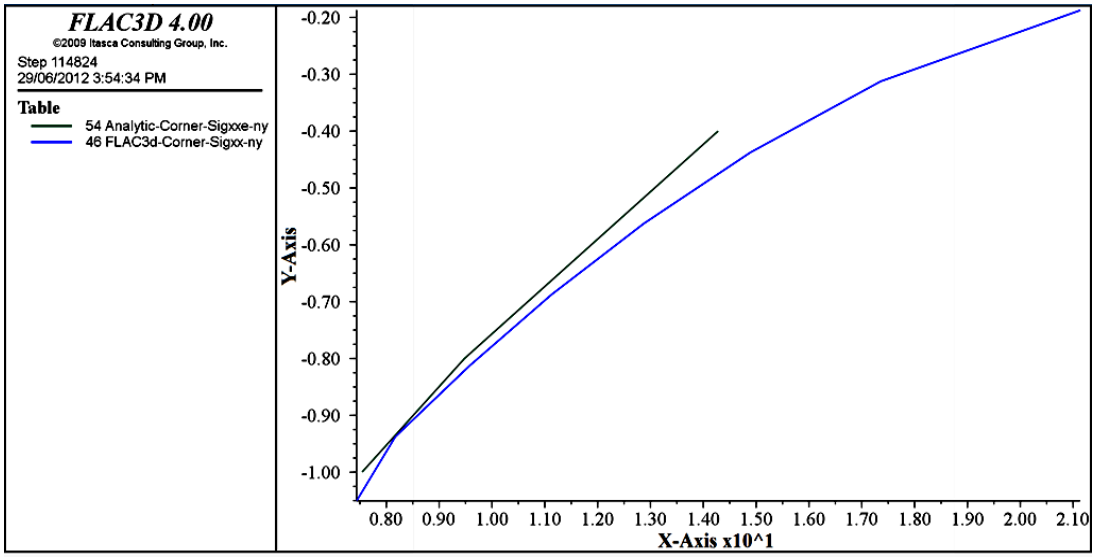


Figure 5.134 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solutions under the corner of a uniform square load in Flac3D sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x)*

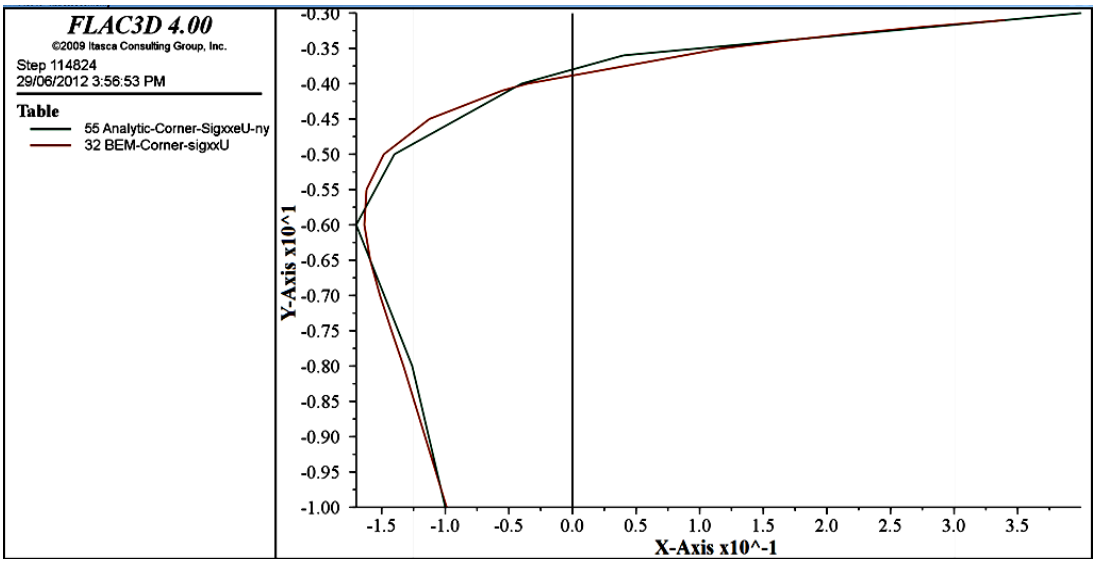


Figure 5.135 Coupled (First Method/IDDM) and exact horizontal stress (σ_x) solutions under the corner of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x)*

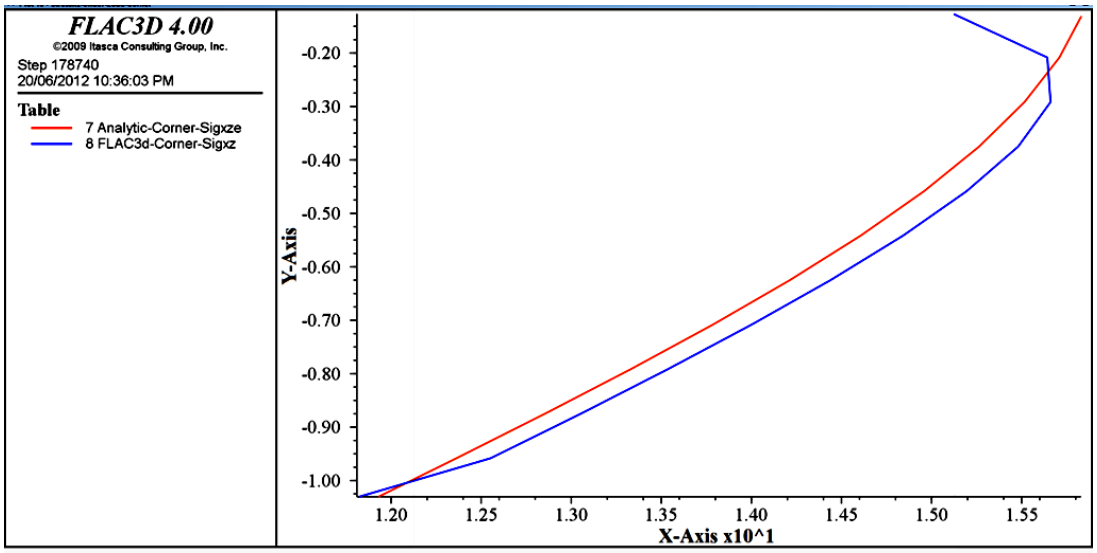


Figure 5.136 Coupled (First Method/IDDM) and exact shear stress (σ_{xz}) solutions under the corner of a uniform square load in Flac3D sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_{xz})*

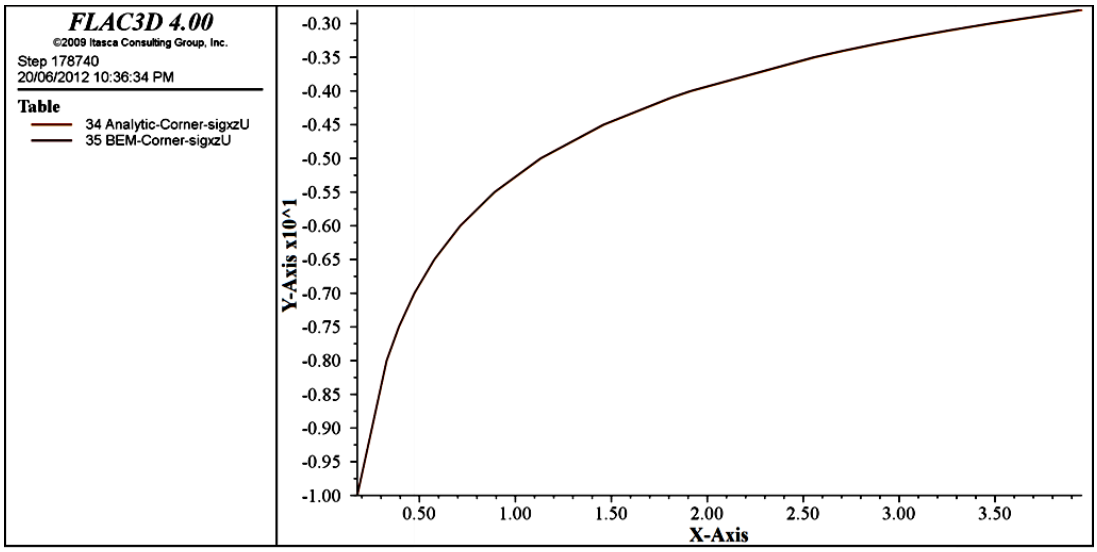


Figure 5.137 Coupled (First Method/IDDM) and exact shear stress (σ_{xz}) solutions under the corner of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_{xz})*

5.3.2.5 Coupled Flac^{3D} -BEM Solution (Second Method/SDDM)

The second coupling method, in the case [υ = 0], corrected the vertical displacement at points A and B, u_z, σ_z and σ_x solutions under the center of the load in the Flac^{3D} sub-domain with significant reduction of error compared to the uncoupled Flac^{3D} solution and with a CPU 67 % less than the runtime required by the first coupling method to solve the same problem (see Tables 5.33-5.35, and Figures 5.138-5.142). The Flac^{3D} model used in this method has the same ratio (R/a) that equals 3 as used before. Tables 5.34, and 5.35 are in Appendix c pp.330-331.

Table 5.33 Vertical displacementof three solutions under a loaded square area, [ν = 0]

Uncoupled Flac^{3D} Solution [coarse mesh]						
Point	**R/a**	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [Center]	3	0.022443994	0.01729818	-22.93	14.527	2048
B [Corner]		0.011221997	0.00653903	-41.73		
Uncoupled Flac^{3D} Solution [fine mesh]						
Point	**R/a**	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [Center]	35	0.022443994	0.021918	-2.34	217.03	34496
B [Corner]		0.011221997	0.01088758	-2.98		
Coupled BEM-Flac^{3D} Solution [First Method/IDDM]						
Point	**R/a**	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [Center]	3	0.02244399	0.02223186	-0.95	134.5 [pre]	2048
B [Corner]		0.01122200	0.01132729	0.94	*5.2[post]*	
Coupled BEM-Flac^{3D} Solution [Second Method/SDDM]						
Point	**R/a**	**Exact u_z [m]**	**Numerical u_z [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [Center]	3	0.02244399	0.02289078	1.99	46.9	2048
B [Corner]		0.01122200	0.01199655	6.90		

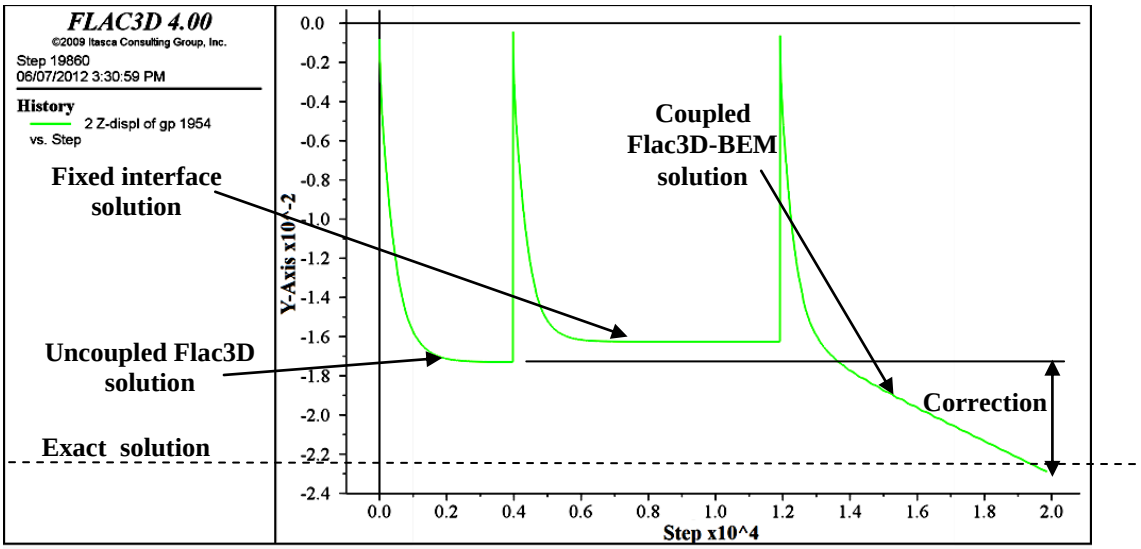


Figure 5.138 History of Flac3D and coupled (Second Method/SDDM) vertical displacement at point A (the center of a uniform square load), ν = 0.0, *(Y-Axis ≡ u_Z)*

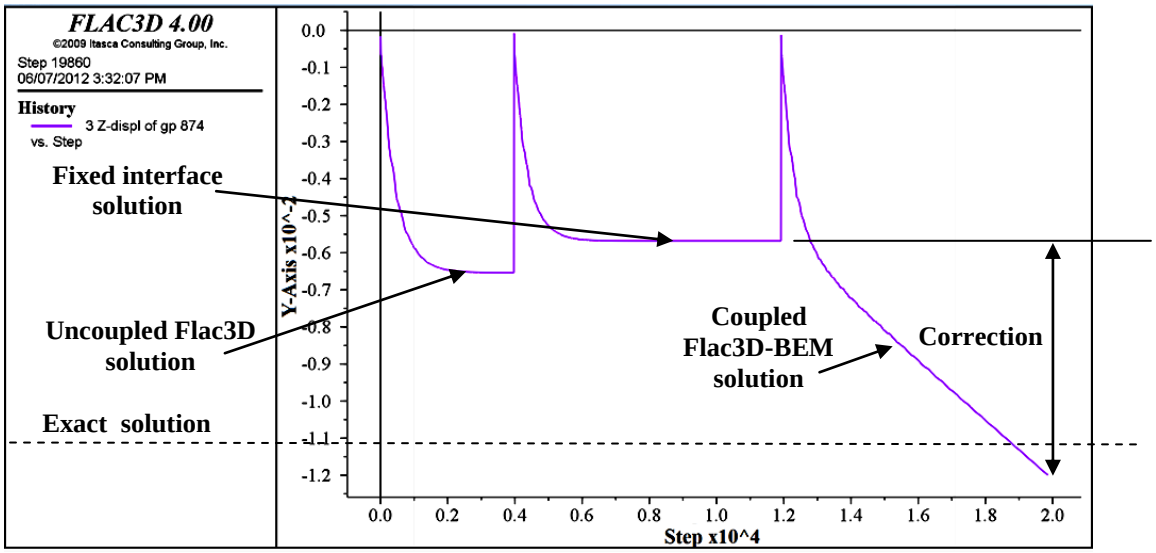


Figure 5.139 History of Flac3D and coupled (Second Method/SDDM) vertical displacement at point B (the corner of a uniform square load), ν = 0.0, *(Y-Axis ≡ u_Z)*.

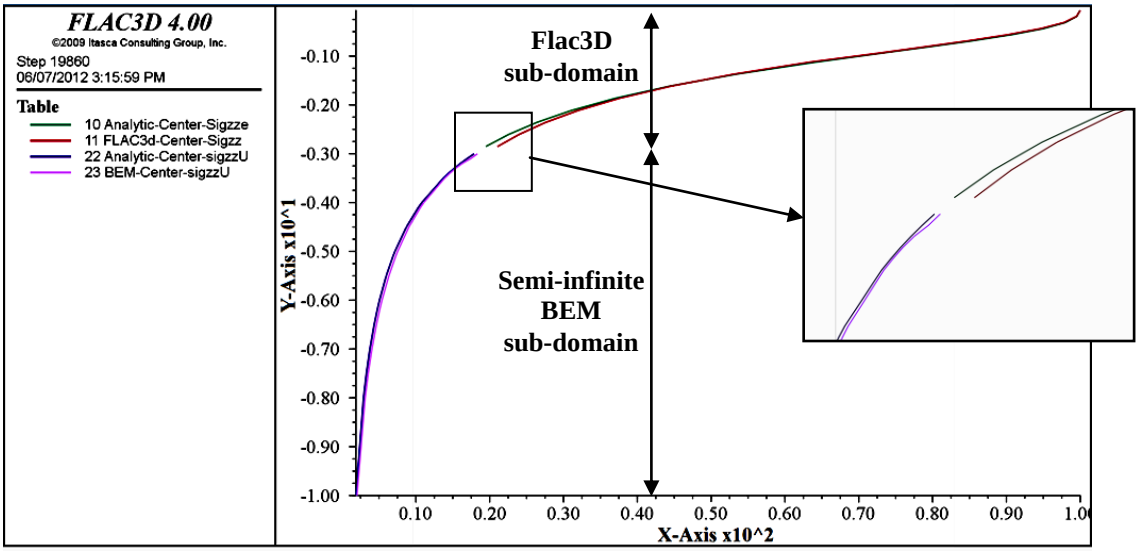


Figure 5.140 Coupled (Second Method/SDDM) and exact vertical stress (σ_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_Z)*

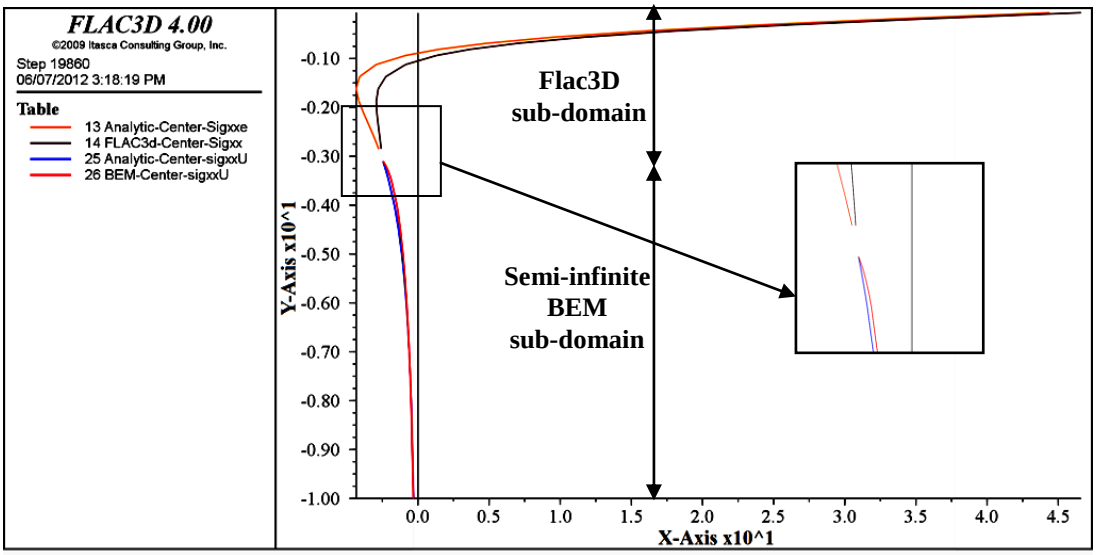


Figure 5.141 Coupled (Second Method/SDDM) and exact horizontal stress (σ_x) solutions under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

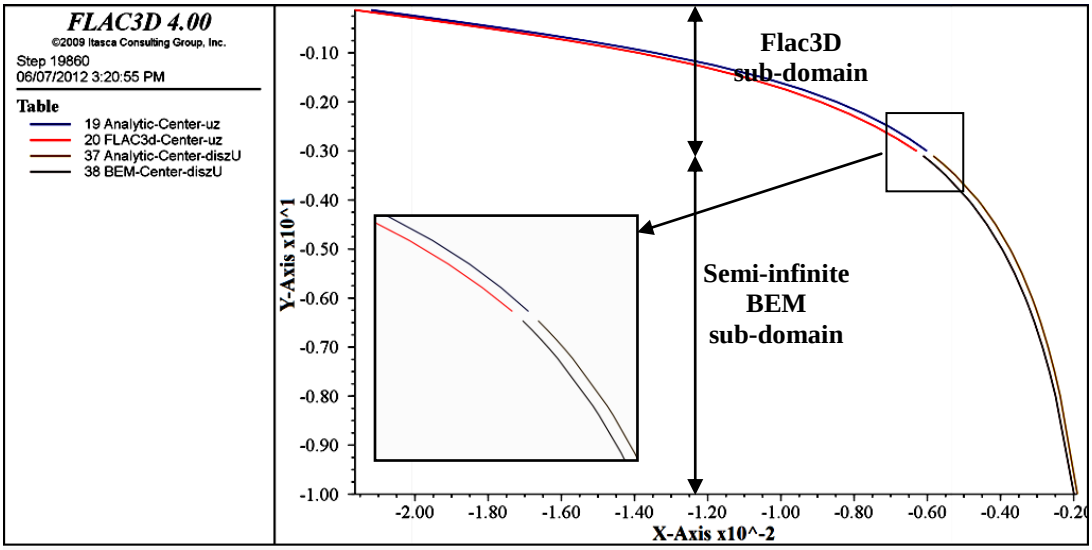


Figure 5.142 Coupled (Second Method/SDDM) and exact vertical displacement (u_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ $_z$).*

The accuracy of the stress solutions (1-10 % RE) and the displacement solution (2-6% RE), created by the Second Method/SDDM, are not as exact compared to the solutions in the First Method/IDDM under the center and the corner of the load in the semi-infinite BEM sub-domain

(see Figures 5.140-5.142 and Figures 5.143, 5.144 in Appendix c, pp.328-329, and Tables 5.34, and 5.35 in Appendix c pp.330-331).

Because the second method solution is a truncation boundary position and Poisson's ratio dependent, a larger Flac^{3D} model of ratio R/a = 4 is used, in the case where Poisson's ratio υ equals 0.3, to reach the same level of accuracy reached in the previous example where R/a equals 3 and υ equals 0.0 (see Tables 5.36-5.38 and Figures 5.145-151). See the Figure 5.151 in Appendix c, p.333 and the Tables 5.37 and 5.38 pp.334-335. The stress continuity and displacement compatibility across the interface are not fully developed in this method (see Figures 5.147-5.149). The effect of Poisson's ratio is clearly visible on the horizontal normal stress $\sigma_{x,}$ under the load corner in BEM sub-domain and close to the interface (see Figure 5.150.c in Appendix c, p.332). The coupled horizontal stress solution's divergence from the exact solution could be corrected by increasing the depth of the interface (larger R/a) and the mesh size as well.

Table 5.36 Vertical displacement's three solutions under a loaded square area, [ν = 0.3]

Uncoupled Flac^{3D} Solution				
Point	**Exact u_z [m]**	**Numerical uz [m]**	**R.E.%**	**Number of zones**
A [Center]	0.020424035	0.01616747	-20.84	2048
B [Corner]	0.010212017	0.00624155	-38.88	
Coupled BEM-Flac^{3D} Solution [First Method]				
Point	**Exact u_z [m]**	**Numerical uz [m]**	**R.E.%**	**Number of zones**
A [Center]	0.02042403	0.02029550	-0.63	2048
B [Corner]	0.01021202	0.01033982	1.25	
Coupled BEM-Flac^{3D} Solution [Second Method]				
Point	**Exact u_z [m]**	**Numerical uz [m]**	**R.E.%**	**Number of zones**
A [Center]	0.02042403	0.02066769	1.19	2048
B [Corner]	0.01021202	0.01076091	5.37	

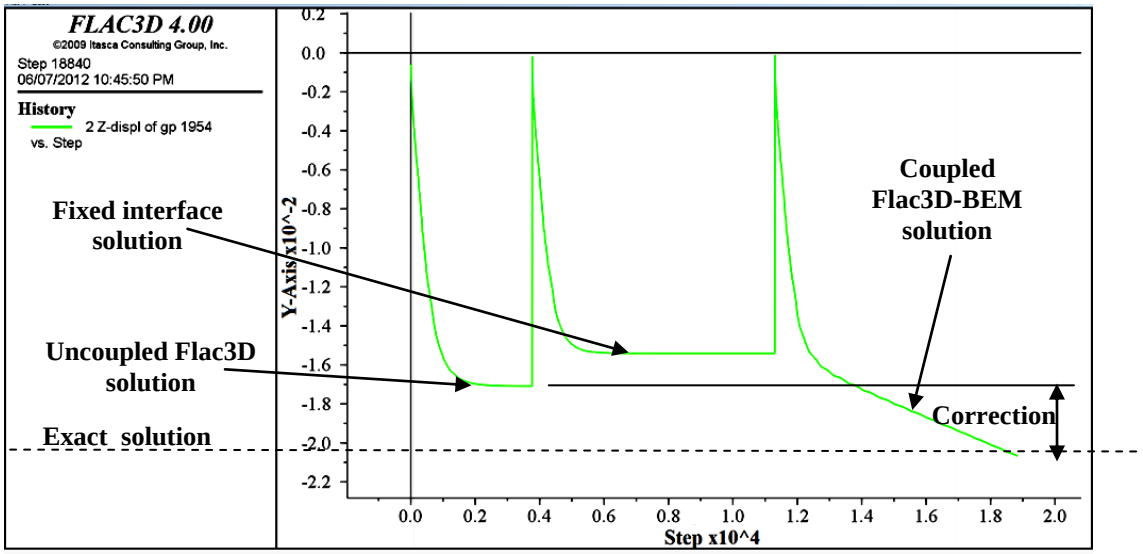


Figure 5.145 History of Flac3D and coupled (Second Method/SDDM) vertical displacement at point A (the center of a uniform square load), ν = 0.3, *(Y-Axis ≡ u_Z)*.

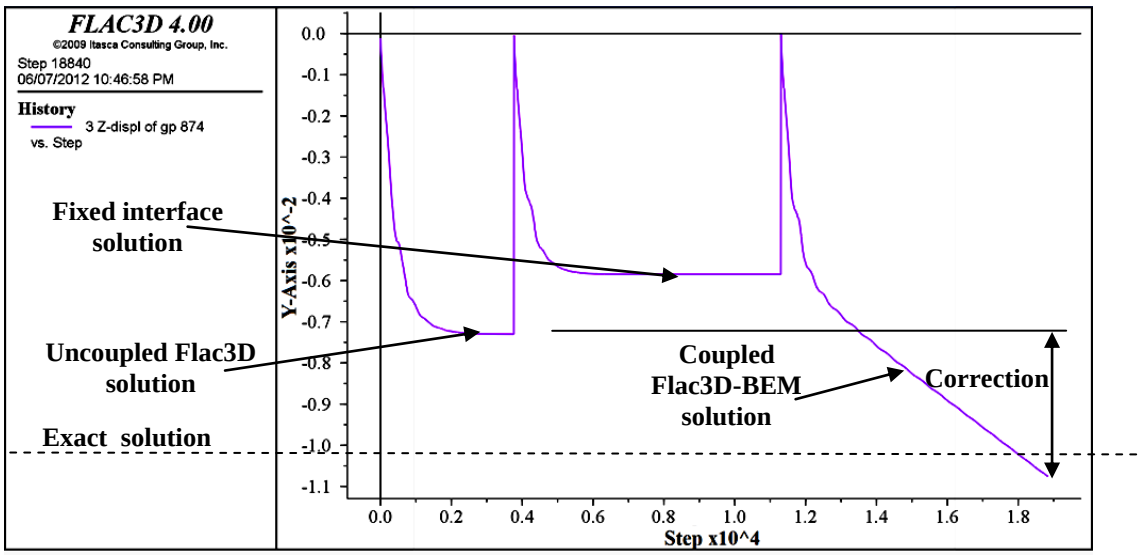


Figure 5.146 History of Flac3D and coupled (Second Method/SDDM) vertical displacement at point A (the center of a uniform square load), ν = 0.3, *(Y-Axis ≡ u_Z)*.

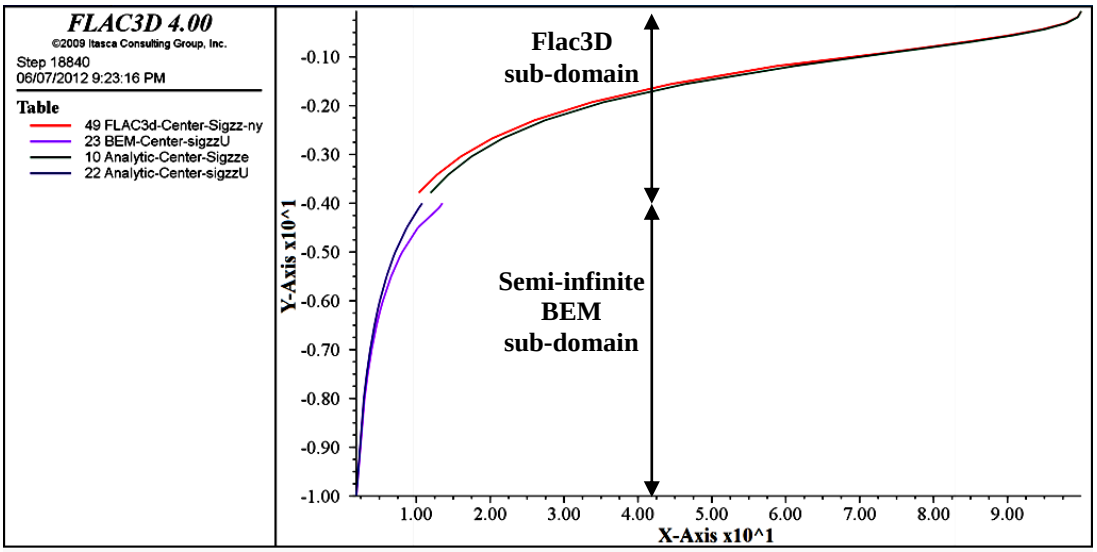


Figure 5.147 Coupled (Second Method/SDDM) and exact vertical stress (σ_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.3,

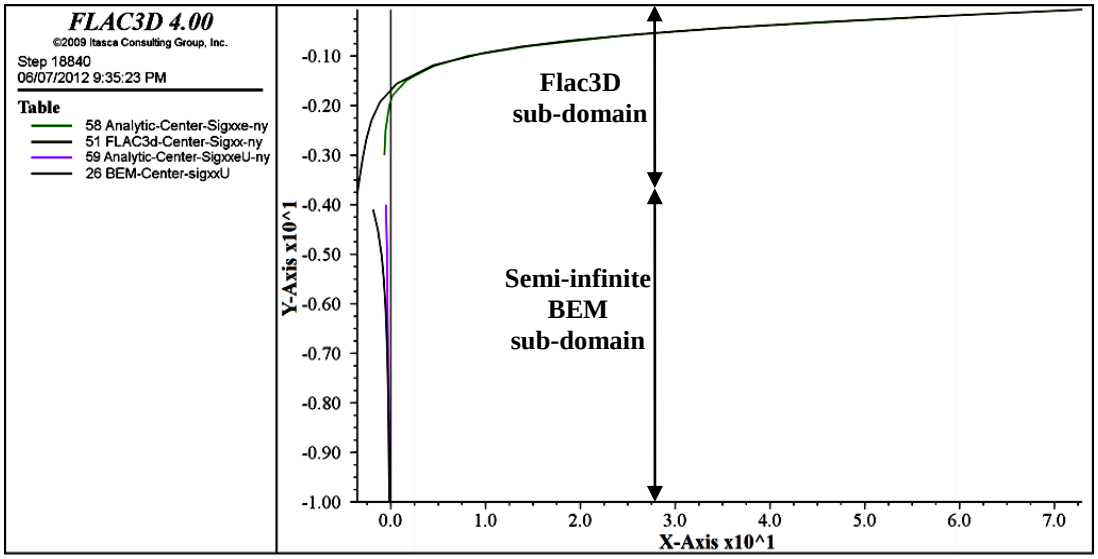


Figure 5.148 Coupled (Second Method/SDDM) and exact horizontal stress (σ_x) solutions under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.3, *(-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

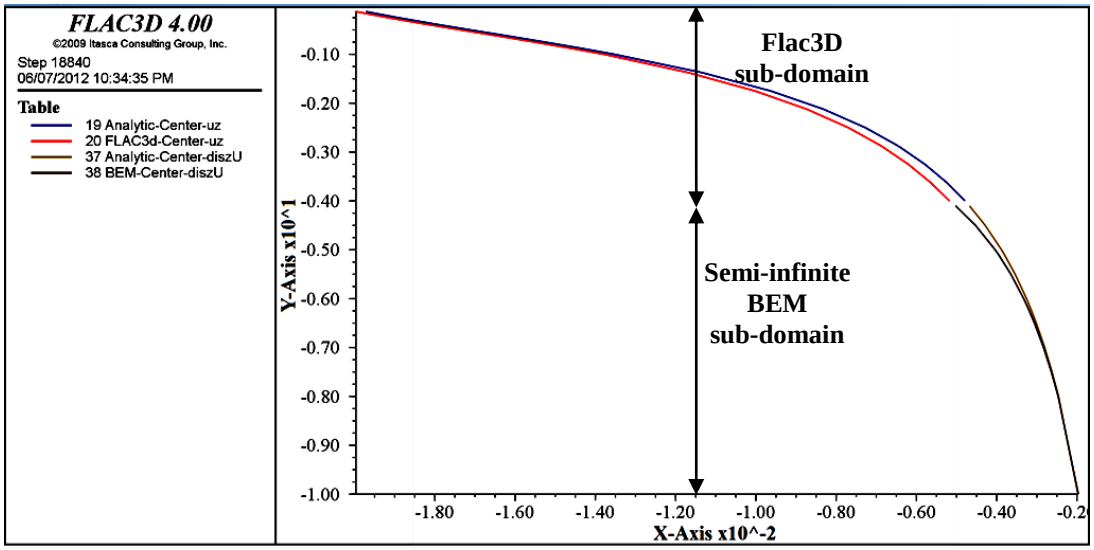


Figure 5.149 Coupled (Second Method/SDDM) and exact vertical displacement (u_z) solution under the center of a uniform square load in both Flac3D and BEM sub-domains, ν = 0.3, *(-Axis ≡ Depth) and (X-Axis ≡ $_z$).*

5.3.3 Cylindrical Tunnel in Infinite Medium

The problem of a hollow cylinder with inner and outer radii, a and b respectively, subjected to uniform internal and external pressure, p_i and p_o respectively, is solved by Lame in 1851 and presented by Timoshenko and Goodier [125] (see Figure 5.152). The radial and tangential stress solutions, σ_r and σ_θ respectively, are given for this plane stress problem through the thickness of the wall of the cylinder by:

$$\sigma_r = \frac{a^2 b^2 (p_o - p_i)}{b^2 - a^2}\frac{1}{r^2} + \frac{p_i a^2 - p_o b^2}{b^2 - a^2} \tag{5.10}$$

$$\sigma_\theta = -\frac{a^2 b^2 (p_o - p_i)}{b^2 - a^2}\frac{1}{r^2} + \frac{p_i a^2 - p_o b^2}{b^2 - a^2}$$

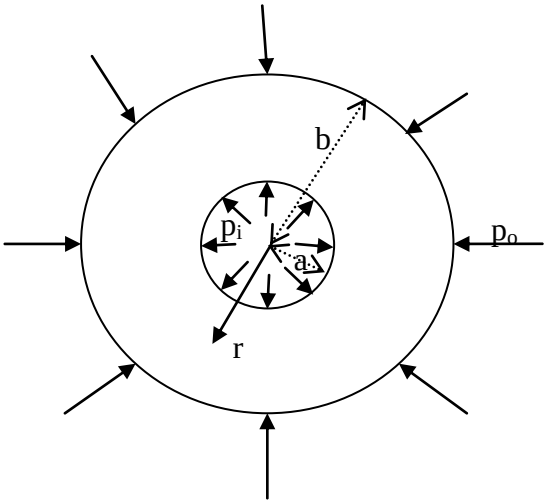


Figure 5.152 Hollow cylinder subjected to internal and external uniform pressure

The plane strain problem of a cylindrical tunnel extended infinitely in an infinite elastic medium is generated from the above hollow cylinder problem assuming $b \to \infty$, p_o=p, p_i = 0 and tunnel radius = a. The radial and tangential stress and radial displacement (u_r) exact solutions in the infinite medium are given by:

$$\sigma_r = -p\left[1-\left(\frac{a}{r}\right)^2\right],\ \sigma_\theta = -p\left[1+\left(\frac{a}{r}\right)^2\right] \quad \text{and} \quad u_r = p\frac{1+\nu}{E}\frac{a^2}{r} \tag{5.11}$$

5.3.3.1 Uncoupled BEM Solution

The fundamental solutions used in equation (2.37) are 2D-Kelvin's fundamental displacement and traction solutions, respectively. Because of unit load applied in k direction in infinite medium for two dimensional plain strain problems, these solutions are given by [5]:

$$\begin{aligned} U_{ki} &= \frac{-1}{8\pi(1-\nu)\mu}\{(3-4\nu)\delta_{ki}\ln r - r_{,k}r_{,i}\} \\ T_{ki} &= \frac{-1}{4\pi(1-\nu)r}\left\{\left[(1-2\nu)\delta_{ki}+2r_{,k}r_{,i}\right]r_{,j}n_{,j} - (1-2\nu)(r_{,k}n_i - r_{,i}n_k)\right\} \end{aligned} \tag{5.12}$$

The infinite medium is subjected to field stress and given in this example the values: σ_z = -1 MPa and σ_x = -1 MPa. The other stress components are zero. Young modulus E=1000 MPa, Poisson's ratio=0.25 and the tunnel radius is a = 1 m. According to the Cauchy rule $t_i = \sigma_{ji} n_j$, the traction components applied on the boundary nodes, are given by:

$$t_x = \sigma_x n_x, t_z = \sigma_z n_z \tag{5.13}$$

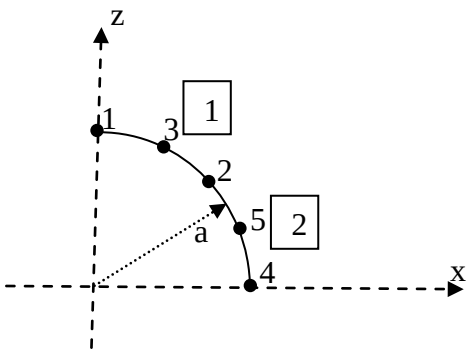


Figure 5.153 Discretization of cylindrical tunnel boundary

The stress components σ_x and σ_z in the vicinity of the tunnel are computed at a number of points (considered as internal points) as a post processing step and the results are superimposed onto the corresponding constant stress field components. The BEM plane strain solution for the circular tunnel problem is in agreement with the analytical solution; the radial displacement u_r (see Figure 5.154), and the radial and tangential stresses σ_r and σ_θ numerical solutions (see Figure 5.155), are accurate (between 1% and 2% RE). This is achieved even with a very coarse boundary mesh, which consists of two one-dimension-second-order boundary elements (see Figure 5.153). The CPU was only less than one second.

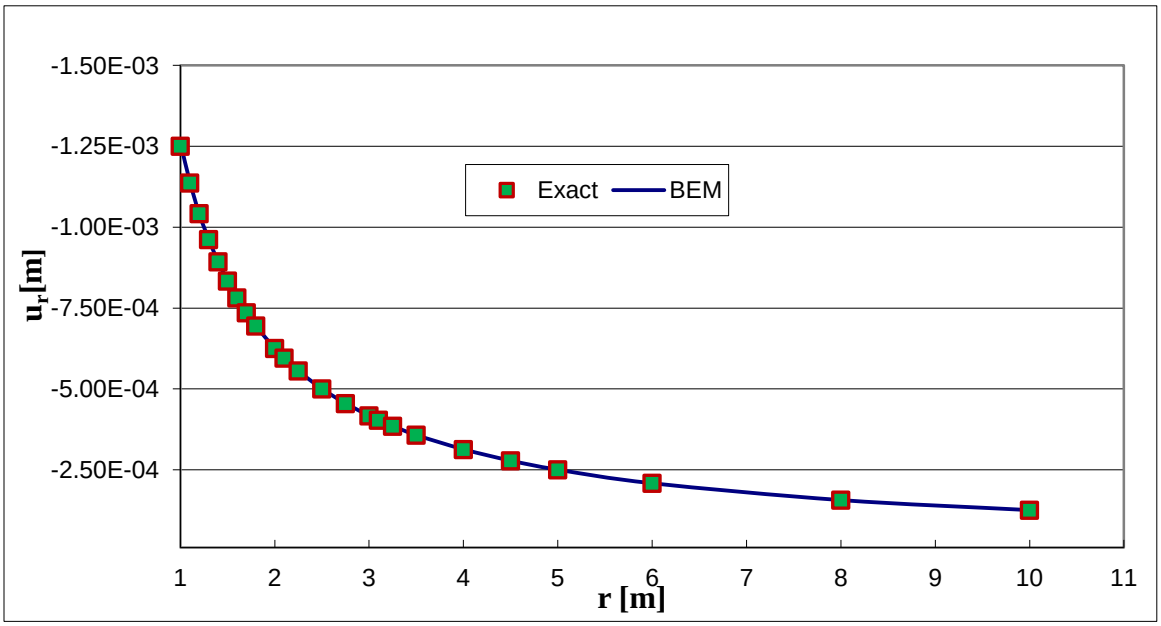


Figure 5.154 Exact and BEM radial displacement distribution in the vicinity of a cylindrical tunnel in infinite medium

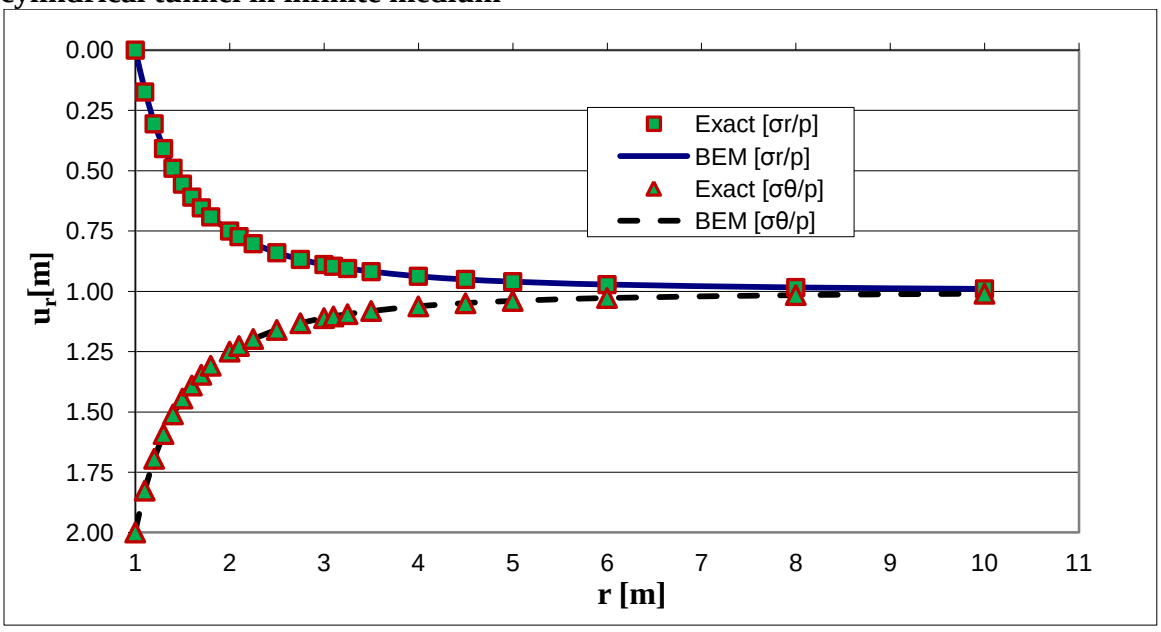


Figure 5.155 Exact and BEM normalized radial and tangential stress nearby a cylindrical tunnel in infinite medium

5.3.3.2 Uncoupled Flac^{3D} Solution

The Flac^{3D} model used to solve this problem is of radius R =2 m or ratio (R/a) = 2 and mesh size or No. of zones = 348. Material properties are E = 1000 MPa and Poisson's ratio = 0.25. Because of the symmetrical nature of the problem in terms of xz and yz planes, only (1/4) of the loaded cylinder is modeled (see Figure 5.156). Two stress components, σ_z = -1 MPa and σ_x = -1 MPa, are initialized in the model and applied to its truncation boundary (Flac^{3D}-BEM sub-domain's interface) as well. The radial displacement (u_r) and the radial and tangential stresses (σ_r and σ_θ) are all overestimated because of the model stressed boundary (see Figures 5.157 and 5.158; and Tables 5.40 and 5.41 in Appendix d, p.336). The Flac^{3D} mesh size and its R/a ratio have been increased significantly to obtain a more accurate solution. Nevertheless, the radial displacement accuracy deteriorated with the increase of r; u_r at point A was accurate, but it was not at point B (see Table 5.39).

Table 5.39 CPU in different numerical solutions for the cylindrical tunnel problem

Uncoupled Flac^{3D} Solution [coarse mesh]						
Point	**R/a**	**Exact u_r [m]**	**Numerical u_r [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [r/a=1]	2	-0.00125	-0.00187	49.81	6.23	384
Uncoupled Flac^{3D} Solution [fine mesh]						
Point	**R/a**	**Exact u_r [m]**	**Numerical u_r [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [r/a=1]	10	0.00125	0.00126	1.19	22.23	1872
B[r/a=10]		0.000125	0.00019	50.48		
Coupled BEM-Flac^{3D} Solution [First Method/IDDM]						
Point	**R/a**	**Exact u_r [m]**	**Numerical u_r [m]**	**R.E.%**	**CPU [Second]**	**Number of zones**
A [r/a=1]	2	-0.00125	-0.00124	-1.12	34.2	384
B[r/a=10]		-0.00013	-0.00012	-1.50		

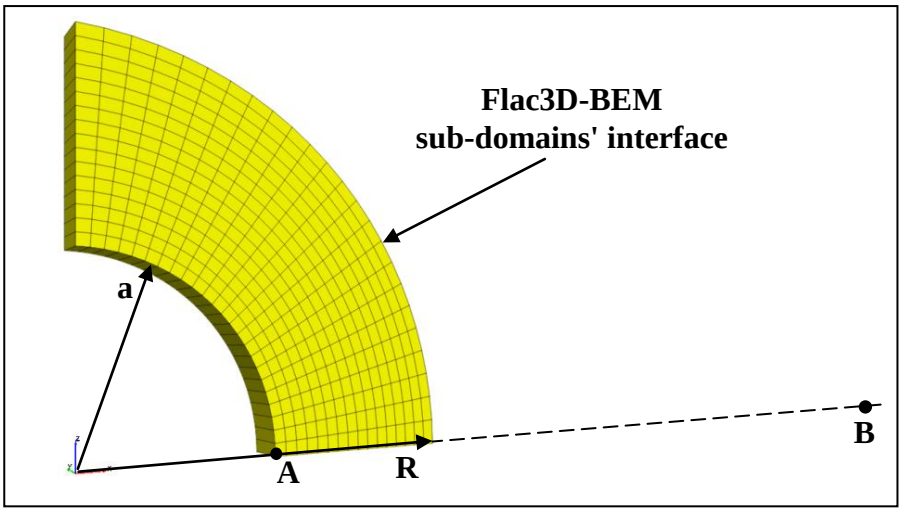


Figure 5.156 Cylindrical tunnel's Flac3D model of ratio (R/a = 2)

Note: In the Flac^{3D} sub-domain, the analytical σ_r is named analytic-sigr, the analytical σ_t is named the analytic-sigt, the numerical σ_r is named Flac^{3D} -sigr and the numerical σ_t is named Flac^{3D} -sigt (see Figure 5.157).

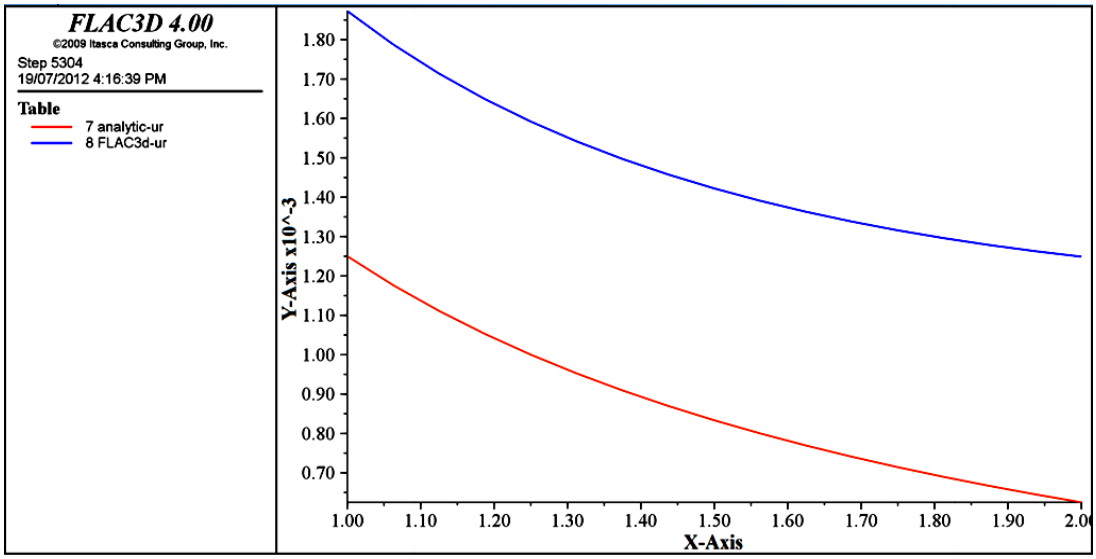


Figure 5.158 Uncoupled Flac3D and exact radial displacement u_r distribution as a function of (r) in the cylindrical tunnel problem, *(Y-Axis ≡ Stress) and (X-Axis ≡ r).*

Note: In the Flac^{3D} sub-domain, the analytical u_r is named analytic-ur and the numerical u_r is named Flac^{3D} -ur (see Figure 5.158).

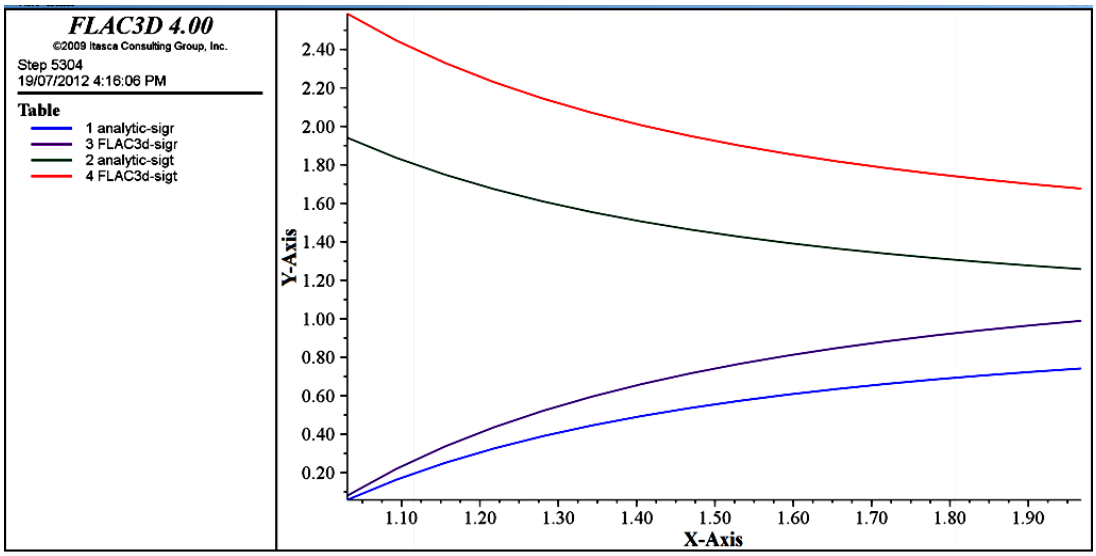


Figure 5.157 Uncoupled Flac3D and exact normalized radial and tangential stress distribution as a function of (r) in the cylindrical tunnel problem , *(Y-Axis ≡ Stress/p) and (X-Axis ≡ r)*

5.3.3.3 Coupled Flac^{3D} -BEM Solution (First Method/IDDM)

A perfect agreement is achieved between the numerical and the exact solutions in the radial displacement and the radial and tangential stresses obtained in both Flac^{3D} and BEM sub-domains after applying the first coupling Flac^{3D} -BEM method. The uncoupled Flac^{3D} solution is corrected by almost 97% and the obtained numerical solution in the BEM sub-domain is almost exact. Furthermore, the stress continuity and the displacement compatibility across the interface are fully developed in this method (see Figures 5.159-5.161). The radial and tangential stresses on the boundary are computed with almost no error, as listed in Tables 5.42 and 5.43 (see Appendix d, pp.337-338). Unlike the uncoupled solution, the radial displacement accuracy in this method is independent from r. Seven iterations are needed in this coupling method where tolerance equals $\epsilon = 0.00006$ and relaxation parameter equals $\omega = 0.4$ to obtain the mentioned corrected solution. The runtimes (CPU) required to obtain both coupled and uncoupled solutions are listed in Table 5.39. Whereas, the coupled method requires more CPU, the

solution accuracy in general is higher and the mechanical responses are computed in the BEM domain with almost no extra cost.

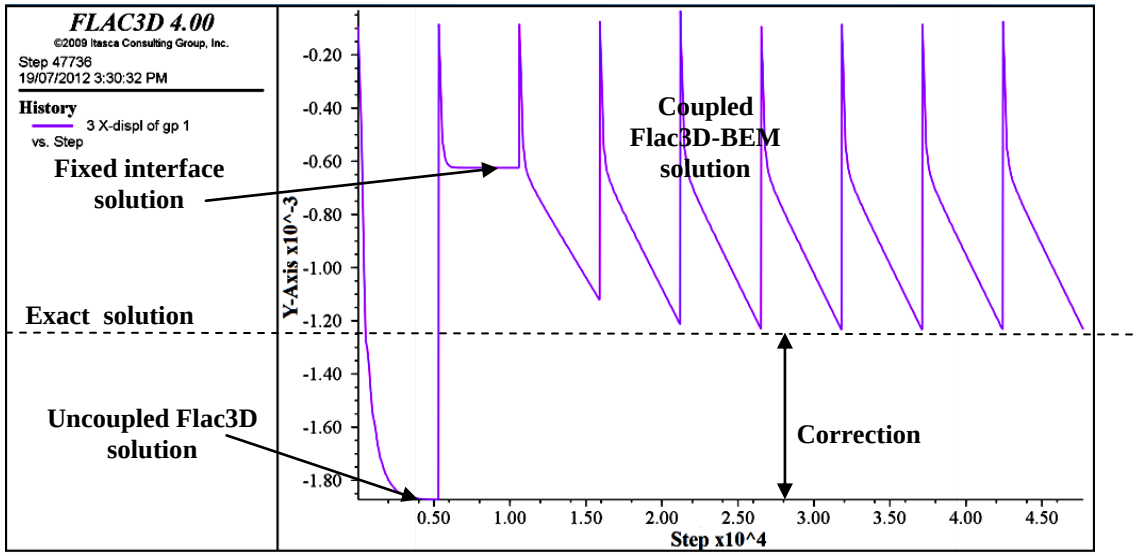


Figure 5.159 History of Flac3D and coupled (First Method/IDDM) displacement ($u_x = u_r$) at point A in the cylindrical tunnel problem, *(Y-Axis ≡ u_Z).*

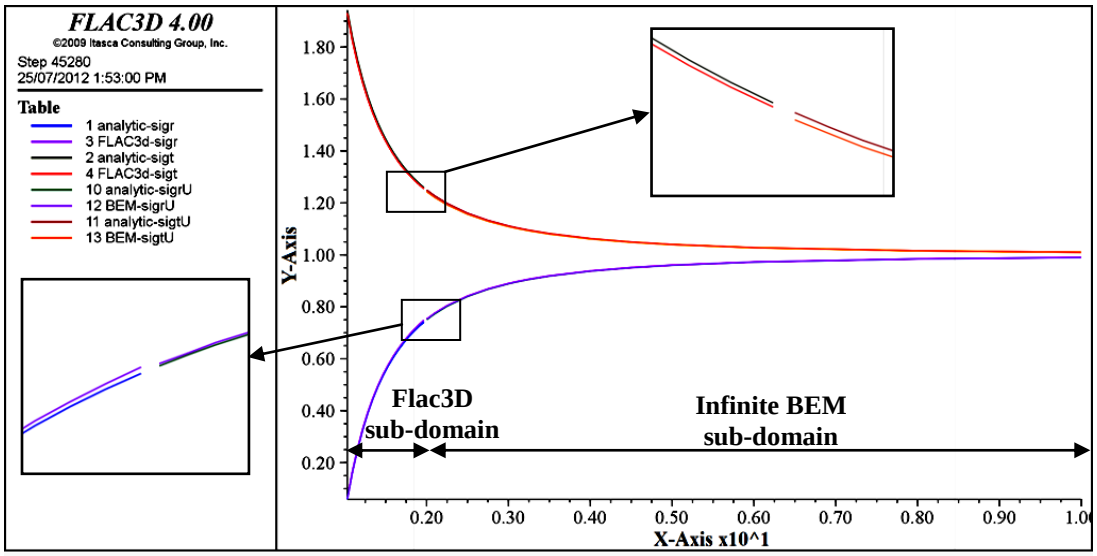


Figure 5.160 Coupled (First Method/IDDM) and exact normalized radial and tangential stress distribution as a function of (r) in both Flac3D and BEM sub-domains, *Y-Axis ≡ Stress/p) and (X-Axis ≡ r).*

Note: In the BEM sub-domain, the analytical σ_r is named analytic-sigrU, the analytical σ_t is named the analytic-sigtU, the numerical σ_r is named BEM-sigrU, and the numerical σ_t is named BEM-sigtU (in Figure 5.160).

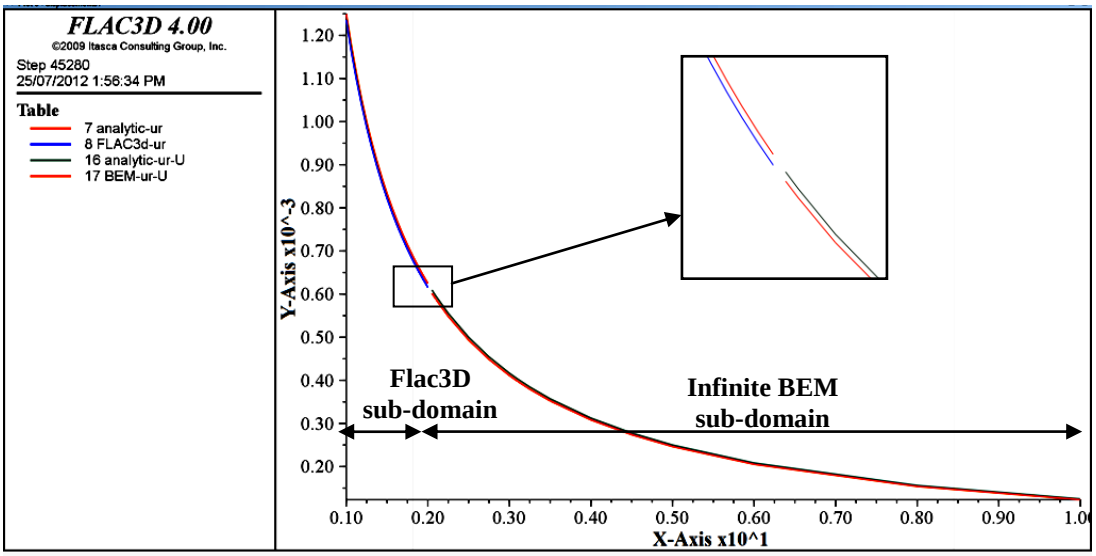


Figure 5.161 Coupled (First Method/IDDM) and exact radial displacement distribution u_r as a function of (r) in both Flac3D and BEM sub-domains, *(Y-Axis ≡ u_r/p) and (X-Axis ≡ r).*

Note: In The BEM sub-domain, the analytical u_r is named analytic-ur-U, and the numerical u_r is named BEM-ur-U (see Figure 5.161).

5.3.3.4 Coupled Flac^{3D} -BEM Solution (Second Method/SDDM)

The upcoming theoretical solution for the cylindrical tunnel problem is following the second coupling method scheme elaborated in section 5.3.1.4. It is used to prove correctness of the method and its dependence on the truncation boundary position and Poisson's ratio value, at the same time. The given relationship between the radial and tangential stresses and the radial and tangential strains in the plane strain problems are [125]:

$$\sigma_r = \frac{E^*}{1-\nu^{*2}}\left[\varepsilon_r + \nu^* \varepsilon_\theta\right] \tag{5.14}$$

$$\sigma_\theta = \frac{E^*}{1-\nu^{*2}}\left[\varepsilon_\theta + \nu^* \varepsilon_r\right] \tag{5.15}$$

$$\text{Where: } \nu^* = \frac{\nu}{1-\nu} \text{ and } E^* = \frac{E}{1-\nu^2}$$

The radial and tangential strains in terms of radial displacement are:

$$\varepsilon_r = \frac{\partial u_r}{\partial r} \text{ and } \varepsilon_\theta = \frac{u_r}{r} \tag{5.16}$$

The governing differential equation of equilibrium for the two dimensional problem, being discussed in this section (where: shear stress $\tau_{r\theta} = 0$), in polar coordinates is:

$$\frac{\partial \sigma_r}{\partial r} + \frac{\sigma_r - \sigma_\theta}{r} = 0 \tag{5.17}$$

If equation (5.16) is substituted in equation (5.17), we get:

$$\frac{\partial u_r^2}{\partial r^2} + \frac{1}{r}\frac{\partial u_r}{\partial r} - \frac{u_r}{r^2} = 0 \tag{5.18}$$

The given general solution for equation (5.18) is:

$$u_r = c_1 r + c_2 \frac{1}{r} \tag{5.19}$$

Where: c_1 and c_2 are integration constants.

Substituting equations (5.16) and (5.19) into equations (5.14) and (5.15) gives:

$$\sigma_r = \frac{E^*}{1-\nu^{*2}}\left[c_1(1+\nu^*) - (1-\nu^*)\frac{c_2}{r^2}\right] \tag{5.20}$$

$$\sigma_\theta = \frac{E^*}{1-\nu^{*2}}\left[c_1(1+\nu^*) + (1-\nu^*)\frac{c_2}{r^2}\right] \tag{5.21}$$

Let's apply to the bounded sub-domain, shown in Figure 5.152, the following boundary conditions according to the second coupling method scheme:

1. The outer boundary at r = b, is fixed (u_r= 0). Substituting this boundary condition in equation (5.19) gives:

$$c_1 = -c_2\frac{1}{b^2} \tag{5.22}$$

2. The radial stress σ_r, which equals (-p) at r = a makes equation (5.20) as:

$$-p = \frac{E^*}{1-\nu^{*2}}[c_1(1+\nu^*) - (1-\nu^*)\frac{c_2}{a^2}] \tag{5.23}$$

and the radial stress at r = b in the bounded sub-domain becomes:

$$\sigma_{r(r=b)} = -\frac{2pa^2}{a^2(1+\nu^*) + b^2(1-\nu^*)} \tag{5.24}$$

The radial displacement at the r =b on the interface of the bounded and infinite sub-domains can be computed from equation (5.11), after replacing (p) with $\sigma_{r(r=b)}$ given in equation (5.24), and replacing tunnel radius (a) with (b), by the following equation:

$$u_{r(r=b)} = -\frac{2p\frac{a^2}{b}(1+\nu^*)}{[\frac{a^2}{b^2}(1+\nu^*) + (1-\nu^*)]E^*} \tag{5.25}$$

The new applied boundary conditions over the bounded domain are:

1. The radial displacement $u_{r(r=b)} = u_o$ at r = b substituted in equation (5.19) gives:

$$c_1 = \frac{u_o}{b} - \frac{c_2}{b^2} \tag{5.26}$$

2. If the radial stress σ_r, which equals (-p) at r = a, is substituted in (5.20) and the initial stress (p) is subtracted from the right side of the equation, the following is obtained:

$$\sigma_r = -\frac{p\left[1-\left(\frac{a}{r}\right)^2\right]b^2}{a^2(1+\nu^*)+b^2(1-\nu^*)}\left[-\frac{u_o E^*}{bp}+(1-\nu^*)\right] \tag{5.27}$$

The ratio of the numerical radial stress solution (Second Method/SDDM) to the exact solution (given in equation 5.11) is:

$$\frac{\sigma_{r[NUM]}}{\sigma_{r[EXACT]}} = -\frac{b^2}{a^2(1+\nu^*)+b^2(1-\nu^*)}\left[-\frac{u_o E^*}{bp}+(1-\nu^*)\right] \tag{5.28}$$

After substituting u_o in equation (5.25), ν^* and E^* in equations (5.14) and (5.15), equation (5.28) becomes:

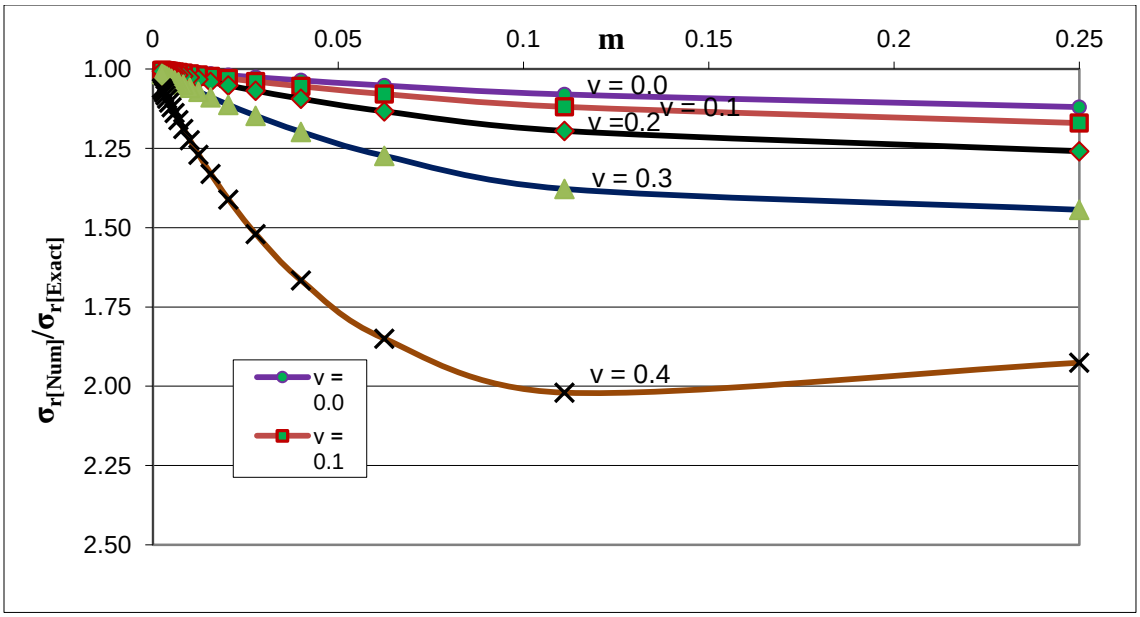
$$\frac{\sigma_{r[NUM]}}{\sigma_{r[EXACT]}} = \frac{1-\nu}{m+1-2\nu}\left[\frac{2m}{m+1-2\nu}+\frac{1-2\nu}{1-\nu}\right] \tag{5.29}$$

Where: $m = \left(\frac{a}{b}\right)^2$

The conclusions reached before in section 5.3.1.4 are confirmed by equation (5.29). The numerical solution is independent from Young modulus (E) and dependent on Poisson's ratio (ν) and truncation boundary position (m). Furthermore, the solution is singular for $\nu = 1$ and it diverges for $\nu = 0.5$ (incompressible material); the farther the truncation boundary is positioned, the larger the stress ratio becomes. Figure 5.162 illustrates that, the higher Poisson's ratio is the farthest the truncation boundary should be positioned, which is the same conclusion as reached before. Nevertheless, the error in the radial stress computed by this method is constant in the

infinite domain regardless of the radial distance r, as could be easily concluded from equation (5.29).

a.



b.

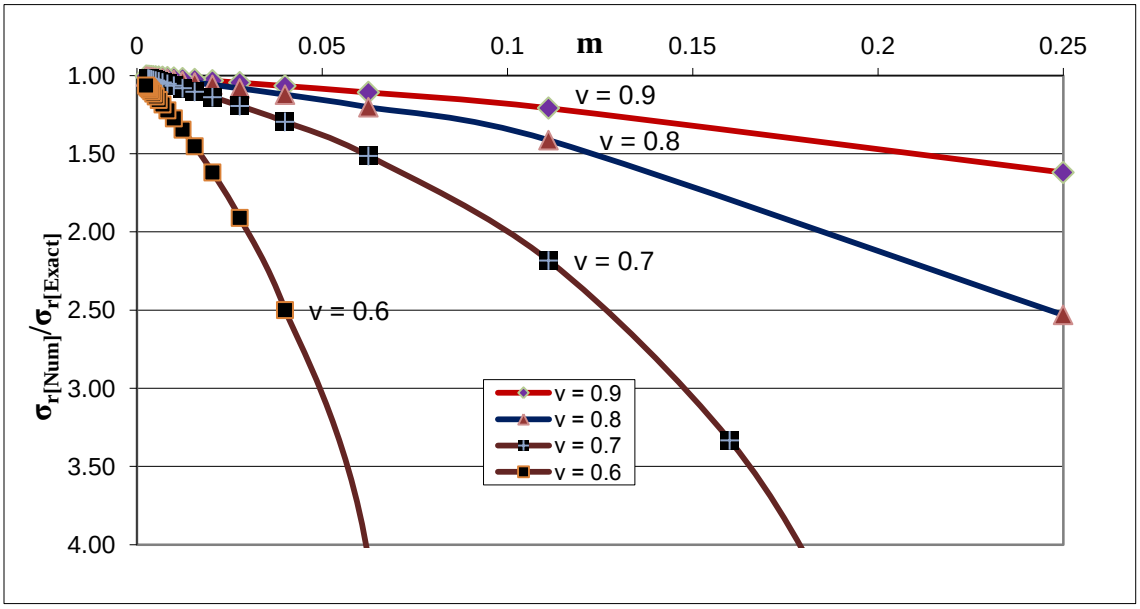


Figure 5.162 Stress ratio ($\sigma_{r[NUM]}/\sigma_{r[EXACT]}$) versus (m) for different Poisson's ratio values

The second coupling method (SDDM) is applied to a Flac^{3D} sub-domain, where Poisson's ratio equals zero and m equals 0.11. The radial stress deviates from the exact solution by almost 8%,

which is the same error ratio, the curves in Figure 5.162 predict (see also Table 5.43). On the other hand, the displacement compatibility and stress continuity across the interface is not completely achieved with this method (see Figures 5.164 and 5.163).

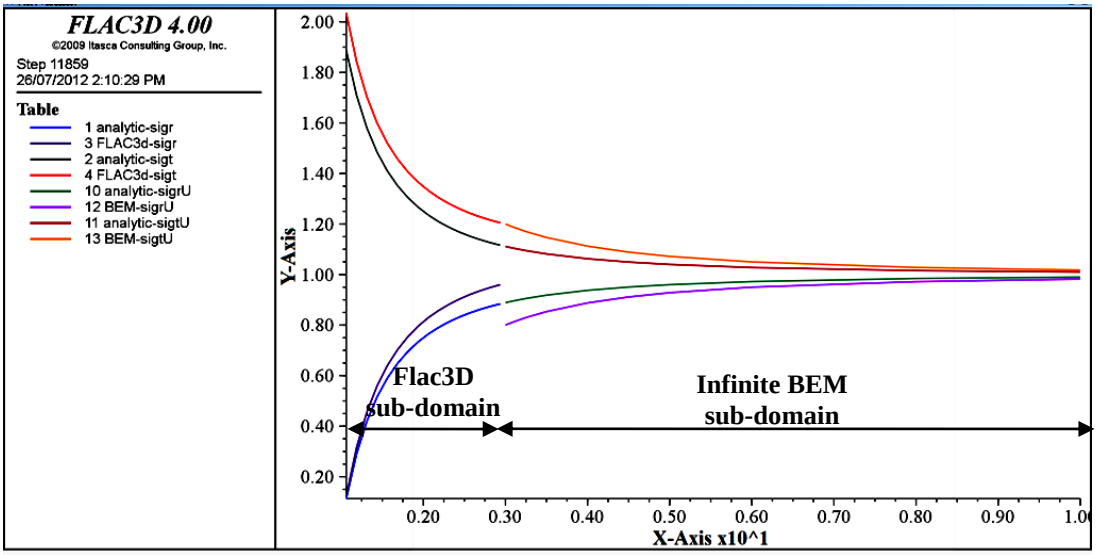


Figure 5.163 Coupled (Second Method/SDDM) and exact normalized radial and tangential stress distribution as a function of (r) in both Flac3D and BEM sub-domains, *(Y-Axis ≡ Stress/p) and (X-Axis ≡ r).*

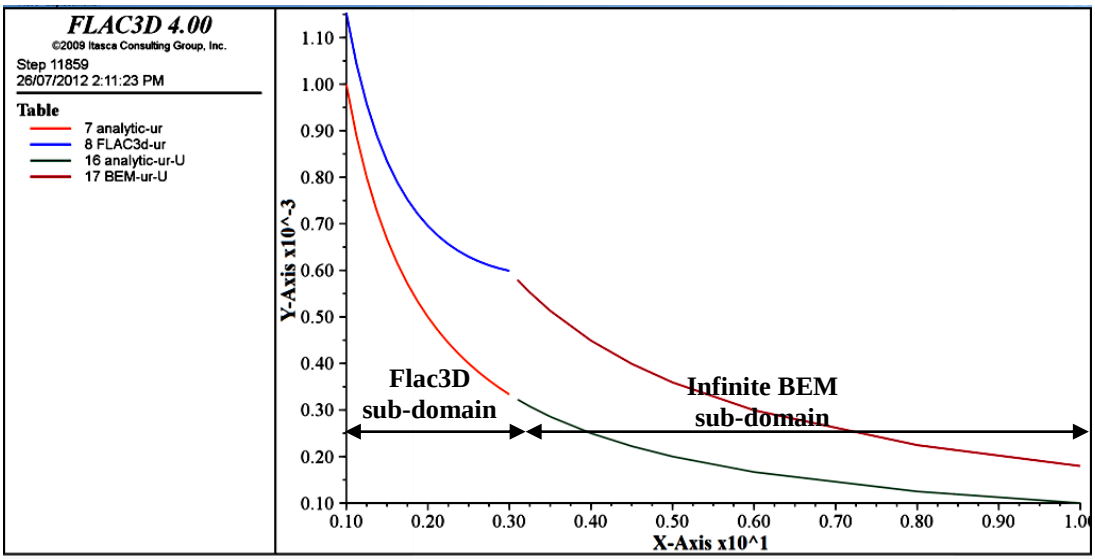


Figure 5.164 Coupled (Second Method/SDDM) and exact radial displacement distribution u_r as a function of (r) in both Flac3D and BEM sub-domains, *(Y-Axis ≡ Stress/p) and (X-Axis ≡ r).*

Table 5.43 Coupled (Second Method/SDDM) normalized radial and tangential stress in an infinite medium in the vicinity of a cylindrical tunnel [m =0.11]

	r	Exact $[\sigma_r/p]$	Numerical $[\sigma_r/p]$	R.E. % in $[\sigma_r]$	Exact $[\sigma_\theta/p]$	Numerical $[\sigma_\theta/p]$	R.E. % in $[\sigma_\theta]$
FLAC^{3D} Sub-domain	1.06	0.1132	0.1240	9.54	1.8868	2.0349	7.85
	1.19	0.2901	0.3152	8.64	1.7099	1.8430	7.78
	1.31	0.4189	0.4548	8.58	1.5811	1.7042	7.78
	1.44	0.5155	0.5590	8.44	1.4845	1.5999	7.78
	1.56	0.5900	0.6394	8.38	1.4100	1.5198	7.78
	1.69	0.6485	0.7029	8.40	1.3515	1.4569	7.80
	1.81	0.6953	0.7536	8.39	1.3047	1.4066	7.81
	1.94	0.7333	0.7954	8.46	1.2667	1.3660	7.84
	2.06	0.7647	0.8296	8.49	1.2353	1.3323	7.85
	2.19	0.7908	0.8577	8.46	1.2092	1.3042	7.85
	2.31	0.8128	0.8816	8.47	1.1872	1.2808	7.88
	2.44	0.8315	0.9024	8.52	1.1685	1.2607	7.89
	2.56	0.8475	0.9201	8.56	1.1525	1.2434	7.89
	2.69	0.8614	0.9351	8.56	1.1386	1.2287	7.91
	2.81	0.8734	0.9483	8.57	1.1266	1.2159	7.93
	2.94	0.8840	0.9599	8.59	1.1160	1.2048	7.96
BEM Sub-domain	3.00	0.8889	0.8002	-9.98	1.1111	1.1998	7.98
	3.10	0.8959	0.8128	-9.28	1.1041	1.1871	7.52
	3.25	0.9053	0.8299	-8.33	1.0947	1.1701	6.89
	3.50	0.9184	0.8534	-7.08	1.0816	1.1466	6.01
	4.00	0.9375	0.8877	-5.31	1.0625	1.1123	4.68
	4.50	0.9506	0.9113	-4.14	1.0494	1.0887	3.75
	5.00	0.9600	0.9282	-3.32	1.0400	1.0718	3.06
	6.00	0.9722	0.9501	-2.27	1.0278	1.0499	2.15
	8.00	0.9844	0.9719	-1.26	1.0156	1.0281	1.22
	10.00	0.9900	0.9820	-0.80	1.0100	1.0180	0.79

5.3.4 A Hole near the Edge of a Semi-infinite Plate under Tension

Mindlin [128] solved the problem of a semi-infinite plate of unit thickness, subjected to uniform tensile stress *T* parallel to its straight edge (plane *y =0*). A cylindrical hole of radius *r'* is situated on *d* depth from the edge with its center on the *y*-axis (see Figure 5.165). The tangential stress components $\overline{\beta\beta}_0 = \sigma_x$ and $\overline{\beta\beta}_1$, along the straight edge and the circular hole edge respectively, are computed and tabulated by Mindlin (in reference [128]) for ten values given to the hole curvature α_1 (from 0.2 to 2.0) associated with ten values of the ratio *d/r'* (from 1.02 to 3.76) respectively.

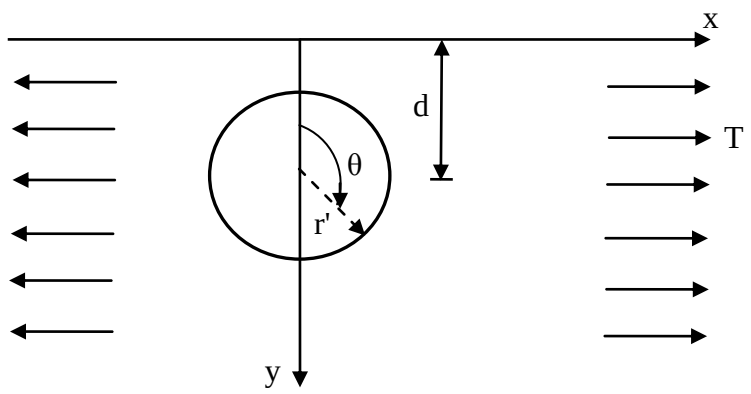


Figure 5.165 Semi-infinite plate with a circular hole and subjected to tension parallel to the plate's straight edge [128].

5.3.4.1 Uncoupled BEM Solution

Equation (2.37) is solved here using 2D-Melan's fundamental displacement and traction solutions, respectively defined in equations (2.83) and their complementary parts in equations (2.84). Because this is a plain stress problem and the mentioned fundamental solutions are derived for plain strain problems, these solutions should be modified by replacing the material properties E and ν by:

$$E^* = E\frac{1+2\nu}{(1+\nu)^2} \quad \text{and} \quad \nu^* = \frac{\nu}{1+\nu} \tag{5.30}$$

The field stress (or the tensile stress T) is given in this example with the numerical value: T = $\sigma_{x\text{-field}}$ = 100 kN/m^2. The other stress components are zero. Mindlin's tangential stress components $\overline{\beta\beta}_0 = \sigma_x$ and $\overline{\beta\beta}_1$, and analytical solutions are independent from material properties E and ν. Nevertheless, numerical methods require specific given numerical values as: Young modulus E=1000 kN/m^2 and Poisson's ratio=0.3. The ratio of the depth of the hole center d to the hole radius r' equals 3.76, associated with the hole curvature α_1 equals 2.0. According to the Cauchy rule $t_i = \sigma_{ji} n_j$, the traction components applied on the boundary nodes, are given by:

$$t_x = Tn_x, t_y = 0 \tag{5.31}$$

Where: $n_x = \sin\theta$

Only the boundary of half of the circular hole (because of the symmetry at the y axis) is discretized into 48 quadratic 3-node-one-dimension equal size boundary elements. The stress $\overline{\beta\beta}_0 = \sigma_x$ on the plate's straight edge is computed at a number of points (considered as internal points) at the post processing step and the results are superimposed onto the constant stress field T (see Table 5.44). Both numerical and exact normalized stress ($\overline{\beta\beta}_0 / T$) are plotted versus (x/r') show agreement (1% RE; see Figure 5.166 and Table 5.44). The required CPU to compute the results was only less than one second..

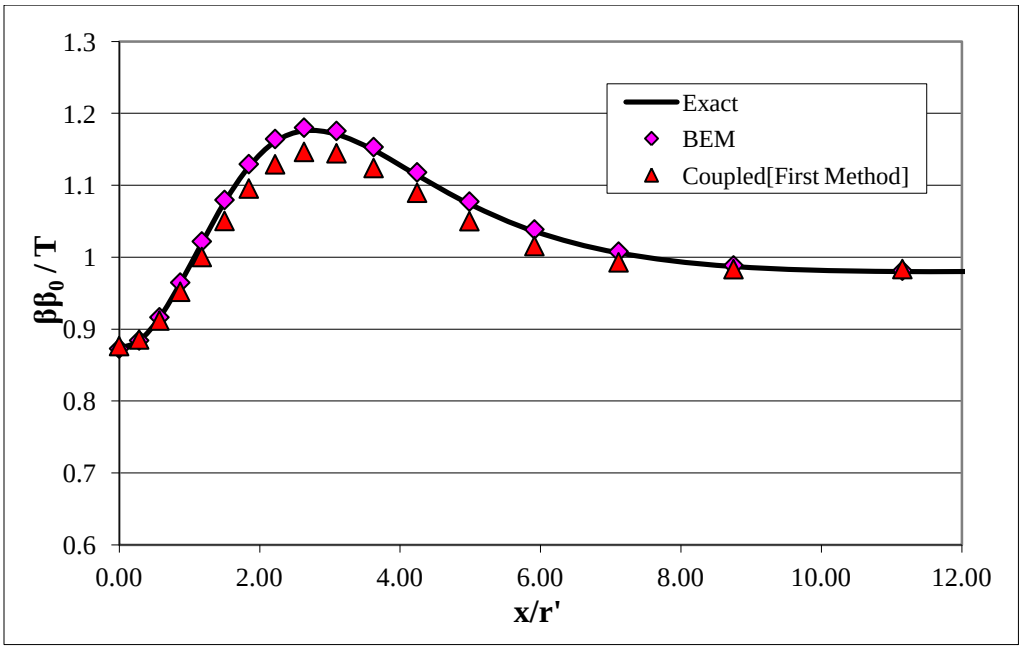


Figure 5.166 Normalized stress along the free traction straight edge of a semi-infinite plate *[d/r' = 3.76, α_1= 2.0]*

Table 5.44 Exact and numerical (BEM and coupled) normalized stress along the free traction straight edge of the semi-infinite plate *[d/r' = 3.76, α_1= 2.0]*

x/r'	Exact $\beta\beta_0/T$	BEM $\beta\beta_0/T$	R.E%	First Coupled Method $\beta\beta_0/T$	R.E%	Second Coupled Method $\beta\beta_0/T$	R.E%
0.0000	0.873	0.8730	0.003	0.8765	0.405	0.9265	6.131
0.2853	0.884	0.8843	0.036	0.8857	0.196	0.9352	5.795
0.5741	0.916	0.9165	0.056	0.9121	-0.428	0.9601	4.815
0.8702	0.964	0.9648	0.083	0.9521	-1.236	0.9978	3.506
1.1777	1.02	1.0220	0.197	1.0004	-1.921	1.0433	2.283
1.5014	1.077	1.0797	0.252	1.0505	-2.457	1.0905	1.257
1.8468	1.126	1.1295	0.313	1.0956	-2.697	1.1331	0.629
2.2211	1.161	1.1645	0.301	1.1293	-2.730	1.1646	0.306
2.6334	1.176	1.1803	0.361	1.1466	-2.503	1.1794	0.285
3.0957	1.171	1.1757	0.402	1.1447	-2.248	1.1736	0.225
3.6246	1.149	1.1533	0.370	1.1242	-2.161	1.1469	-0.181
4.2438	1.114	1.1181	0.372	1.0898	-2.174	1.1039	-0.902
4.9888	1.074	1.0775	0.324	1.0502	-2.213	1.0549	-1.776
5.9148	1.036	1.0387	0.259	1.0156	-1.967	1.0123	-2.285
7.1136	1.006	1.0079	0.186	0.9930	-1.295	0.9850	-2.085
8.7505	0.987	0.9886	0.166	0.9835	-0.357	0.9745	-1.270
11.1553	0.98	0.9814	0.141	0.9836	0.371	0.9761	-0.397
15.0975	0.983	0.9835	0.053	0.9886	0.565	0.9836	0.063
22.8847	0.99	0.9905	0.049	0.9943	0.434	0.9919	0.192
46.0547	0.997	0.9973	0.026	0.9985	0.150	0.9979	0.087

5.3.4.2 Coupled Flac^{3D} -BEM Solution *[d/r' = 3.76 and α_1= 2.0]*

The Flac^{3D} model used to solve this problem is of ratio (R/r') = 2.64 and mesh size equals 768 zones (see Figure 5.167, and Table 5.45). However, the model represents only the cylinder around the hole, but it does not reach the level of the plate's straight edge. The used material properties, d/r', α_1 and the tensile stress T are assigned the same numerical values given before in the BEM solution. The stress component, σ_x = T is initialized in the model and applied to its truncation boundary (Flac^{3D} -BEM sub-domain's interface) as well. The normalized stress $\overline{\beta\beta}_0 / T$ along the straight edge, which cannot be computed by using the uncoupled Flac^{3D} model, is obtained either by the first (IDDM) or the second (SDDM) coupling methods, with agreement of 1% to 6% RE with the exact solution (see Table 5.44; and Figures 5.166, 5.168, and 5.169).

Table 5.45 Normalized stress at points C and D (the stress peak) on the semi-infinite plate's straight edge computed by three different numerical methods *[d/r'= 3.76, α_1= 2.0]*

Uncoupled Flac^{3D} Solution						
Point	**R/r'**	**Exact $\beta\beta_0$/T**	**Numerical $\beta\beta_0$/T**	**R.E.%**	**CPU [Second]**	**Number of zones**
C [x/r'=0]	N/A	0.873	0.937	7.38	25.7	3360
D[x/r'=2.6]		1.176	1.173	-0.28		
Coupled BEM-Flac^{3D} Solution [first method/IDDM]						
Point	**R/r'**	**Exact $\beta\beta_0$/T**	**Numerical $\beta\beta_0$/T**	**R.E.%**	**CPU [Second]**	**Number of zones**
C [x/r'=0]	2.64	0.873	0.8765	0.41	22.8	768
D[x/r'=2.6]		1.176	1.1466	-2.50		
Coupled BEM-Flac^{3D} Solution [second method/SDDM]						
Point	**R/r'**	**Exact $\beta\beta_0$/T**	**Numerical $\beta\beta_0$/T**	**R.E.%**	**CPU [Second]**	**Number of zones**
C [x/r'=0]	2.64	0.873	0.9265	6.13	10.0	768
D[x/r'=2.6]		1.176	1.1794	0.29		

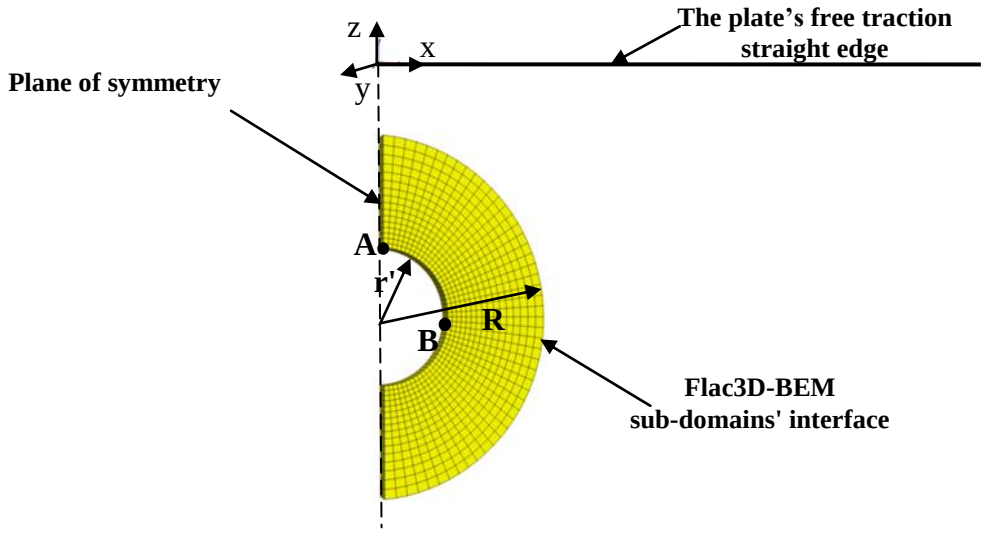


Figure 5.167 A Flac3D model of ratio (R/r' = 2.64) for a semi-infinite plate subjected to tensile stress with a hole near its straight edge, *[d/r'= 3.76, α_1= 2.0]*

Six iterations are needed in the first coupling method (IDDM) where tolerance ϵ equals 0.0006 and relaxation parameter ω equals 0.4 to obtain the mentioned stress solution along the plate's free traction straight edge.

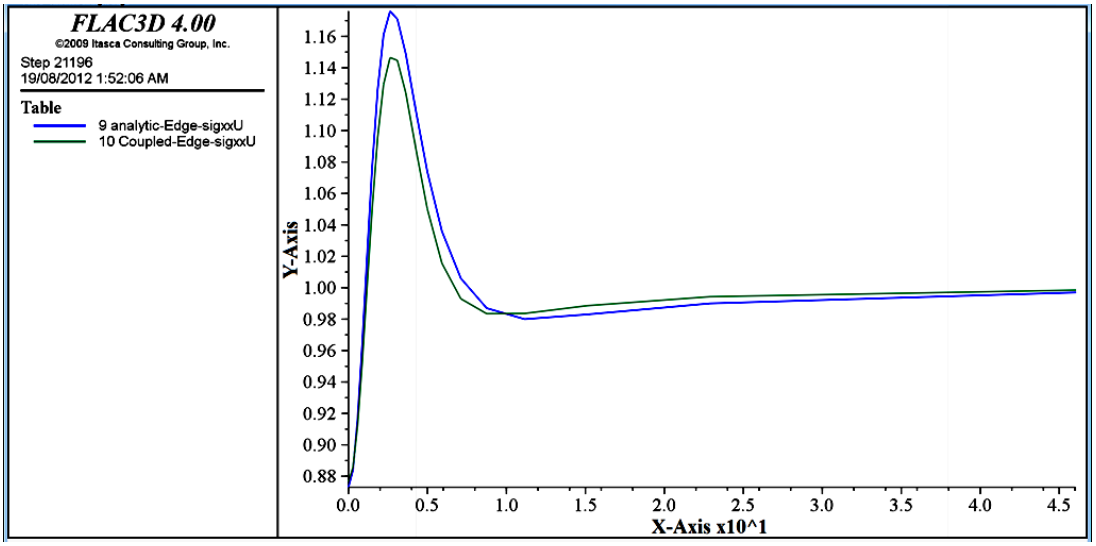


Figure 5.168 Coupled (First Method/IDDM) and exact normalized stress along the free traction straight edge of a semi-infinite plate *[d/r'= 3.76, α_1= 2.0], (Y-Axis ≡ $\beta\beta_0/T$) and (X-Axis ≡ x/r').*

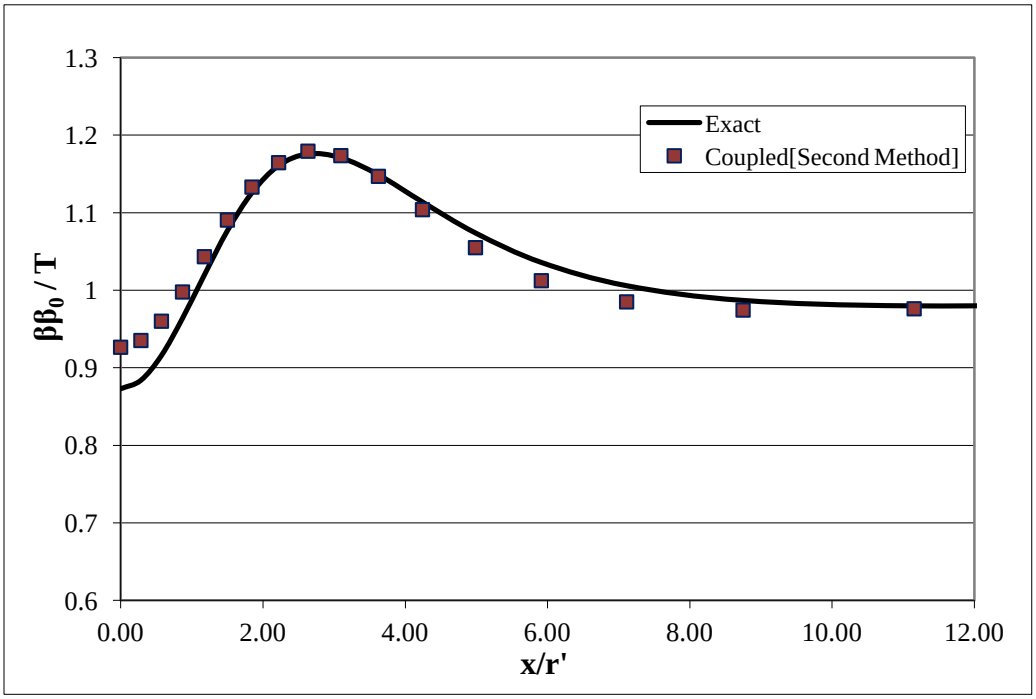


Figure 5.169 Coupled (Second Method/SDDM) and exact normalized stress along the free traction straight edge of a semi-infinite plate *[d/r'= 3.76, α_1= 2.0]*

The coupled solutions of vertical displacement at point A and horizontal displacement at point B using the first method are observed to be between the uncoupled and fixed interface solutions. On the other hand, both coupled solutions by the first method are converging into unknown but predictable exact solutions (see Figure 5.170).

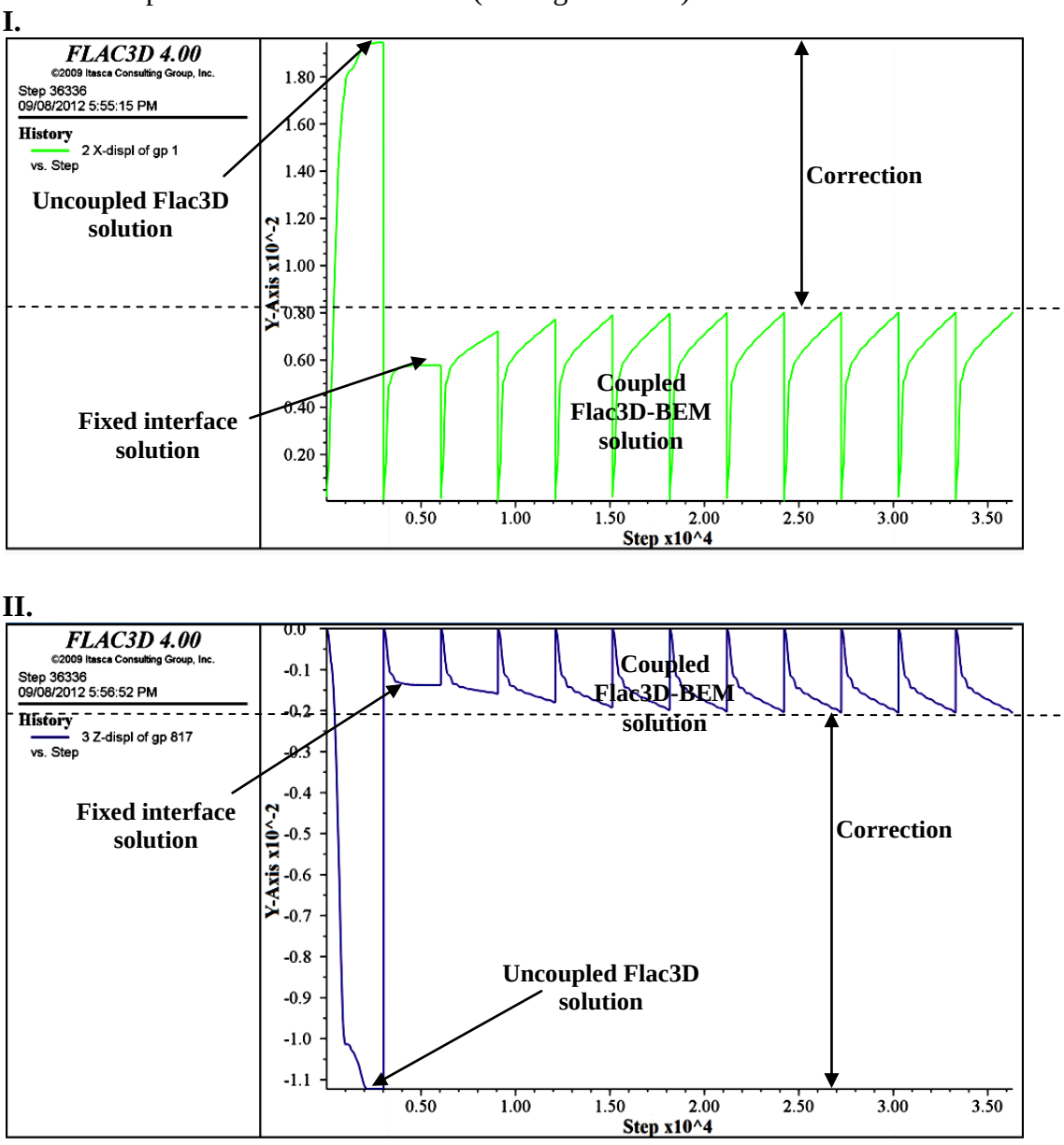


Figure 5.170 History of Flac3D and coupled (First Method/IDDM) displacement at: I. Point B, *(Y-Axis ≡ u_x);* II. Point A, *(Y-Axis ≡ u_z).* At the hole edge in the semi-infinite plate, *[d/r'= 3.76, α_1= 2.0].*

Uncoupled solution with a different Flac^{3D} model configuration *[d/r'= 3.76, α_1= 2.0]:*

To compare, the coupled with the uncoupled stress results (see Figure 5.172) along the plate's free traction straight edge, a different Flac^{3D} model configuration is created. This model not only represents a big area of the medium that surrounds the hole, but also the plate's straight edge (see Figure 5.171). Although, the stress average accuracy at points C (at the plane of symmetry) and D (at the stress peak) is the highest in the first coupling method compared to the second coupling method and the uncoupled method, but the required CPU in the second method is the lowest (see Table 5.45).

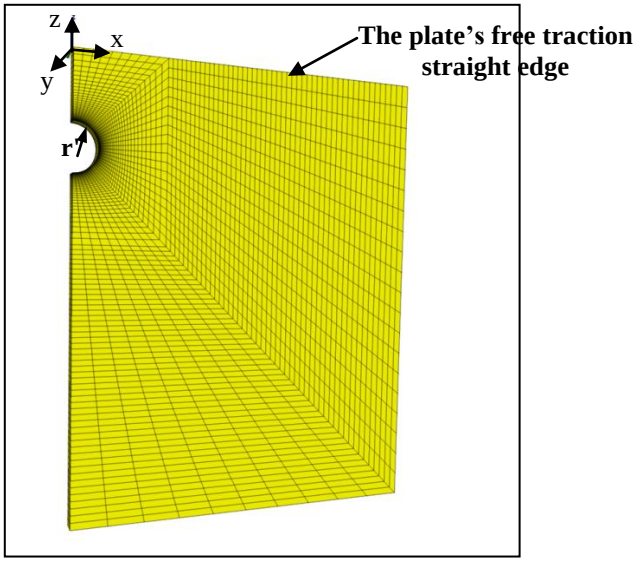


Figure 5.171 A different Flac3D model configuration for the semi-infinite plate subjected to tensile stress with a hole near its straight edge, *[d/r'= 3.76, α_1= 2.0]*

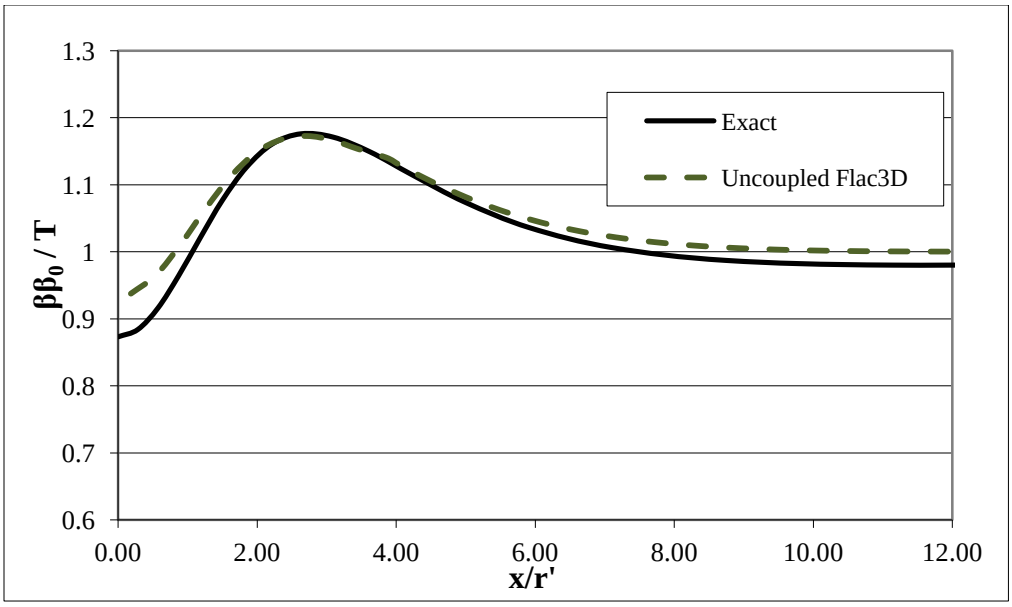


Figure 5.172 Uncoupled and exact normalized stress along the free traction straight edge of a semi-infinite plate *[d/r'= 3.76, α_1= 2.0]*

5.3.4.2 Coupled Flac^{3D} -BEM Solution *[d/r'= 10, α_1= 2.0]*

The uncoupled Flac^{3D} solution becomes more costly, in terms of the number of zones and/or building a mesh with suitable configuration if the depth of the hole's center increases significantly. In other words, obtaining an accurate uncoupled stress solution along the plate's straight edge demands a refined mesh (the center of zones at the plate's edge should be close enough to the edge), which leads to an increasing mesh size by increasing the hole depth. A numerical example of ratio d/r' equals 10 is solved using the three coupled and uncoupled methods, and comparing them with the BEM solution (see the Flac^{3D} model used in Figure 5.173). The accuracy of the stress solution along the plate's straight edge in the second coupling method is the highest (see Tables 5.46 and 5.47, compared to the first coupling method solution in Figure 5.174, and the uncoupled solution in Figure 5.175). Furthermore, the required runtime is the least (see Table 5.46). This accuracy is achieved because Poisson's ratio has no effect on the plate's edge stress as Mindlin's analytical solution for this problem proved. Conversely, the

required CPU to obtain the uncoupled solution is the highest and its accuracy is the least compared to the other solutions.

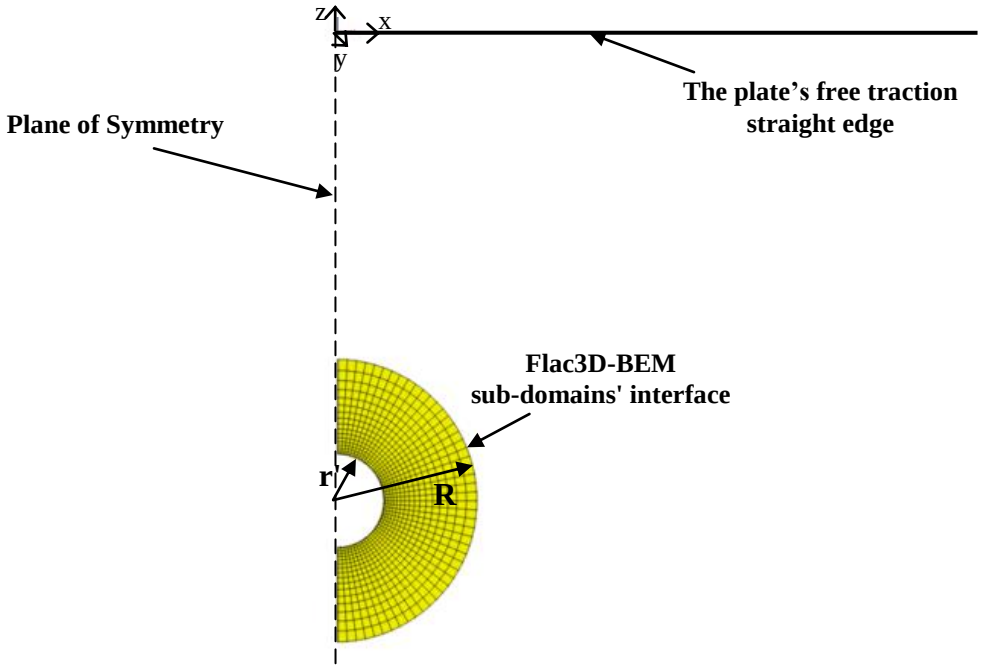
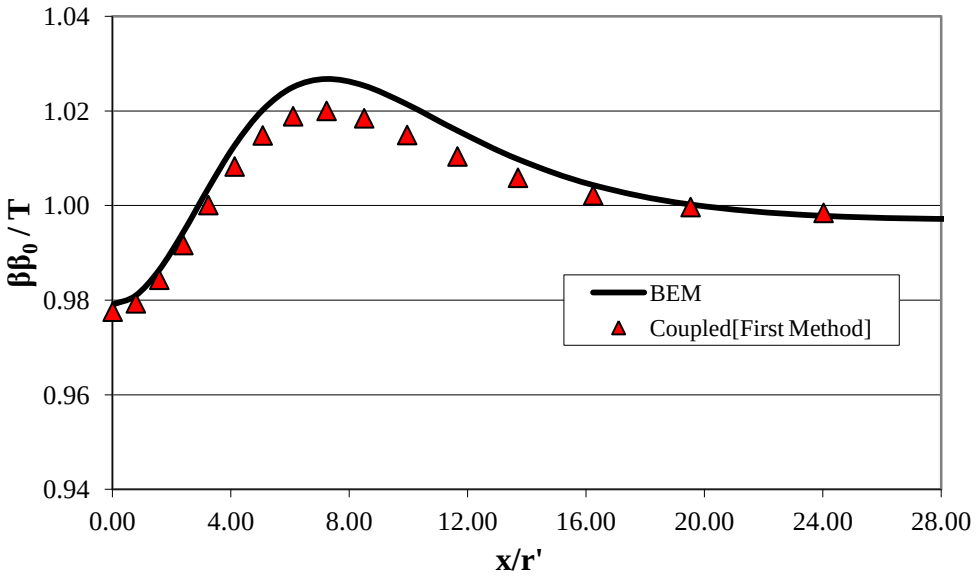


Figure 5.173 A Flac3D model of ratio (R/r' = 3.0) for the semi-infinite plate subjected to tensile stress with a hole near its straight edge, *[d/r'= 10, α_1= 2.0]*

Table 5.46 Normalized stress at points C and D (the stress peak) on the semi-infinite plate's straight edge computed by three different numerical methods *[d/r'= 10.0, α_1= 2.0]*

Uncoupled Flac^{3D} Solution						
Point	**R/r'**	**BEM $\beta\beta_0$/T**	**Coupled $\beta\beta_0$/T**	**R.D.%**	**CPU [Second]**	**Number of zones**
C [x/r'=0]	N/A	0.9791	0.9847	0.58	42.6	7560
D[x/r' = 7.2]		1.0268	1.0249	-0.18		
Coupled BEM-Flac^{3D} Solution [First Method/IDDM]						
Point	**R/r'**	**BEM $\beta\beta_0$/T**	**Coupled $\beta\beta_0$/T**	**R.D.%**	**CPU [Second]**	**Number of zones**
C [x/r'=0]	3.00	0.9791	0.9776	-0.154	26.7	768
D[x/r' = 7.2]		1.0268	1.0200	-0.661		
Coupled BEM-Flac^{3D} Solution [Second Method/SDDM]						
Point	**R/r'**	**BEM $\beta\beta_0$/T**	**Coupled $\beta\beta_0$/T**	**R.D.%**	**CPU [Second]**	**Number of zones**
C [x/r'=0]	3.00	0.9791	0.9752	-0.391	13.9	768
D[x/r' = 7.2]		1.0268	1.0268	0.004		

I.



II.

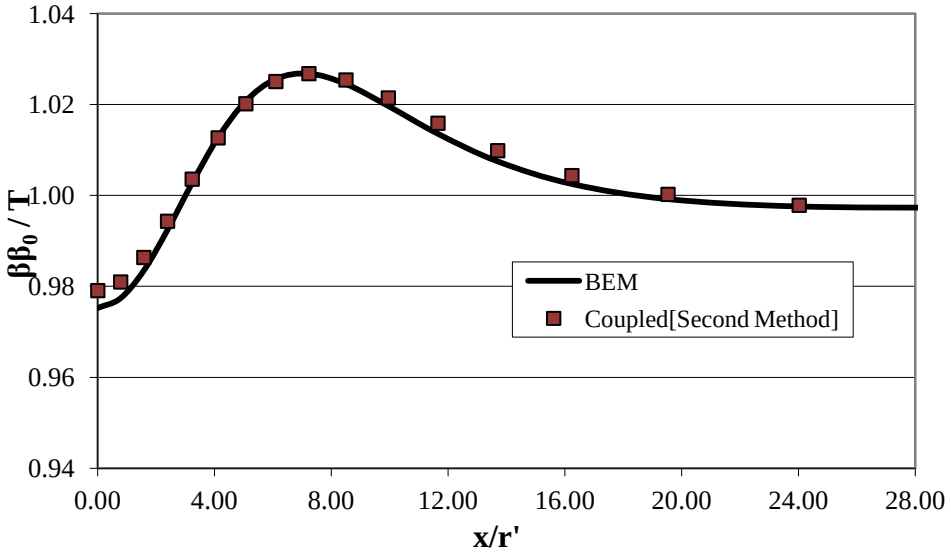


Figure 5.174 Coupled and exact normalized stress along the free traction straight edge of a semi-infinite plate: I. Coupled Flac3D-BEM (First Method/IDDM); II. Coupled Flac3D-BEM (Second Method/SDDM). *[d/r'= 10, α_1= 2.0]*.

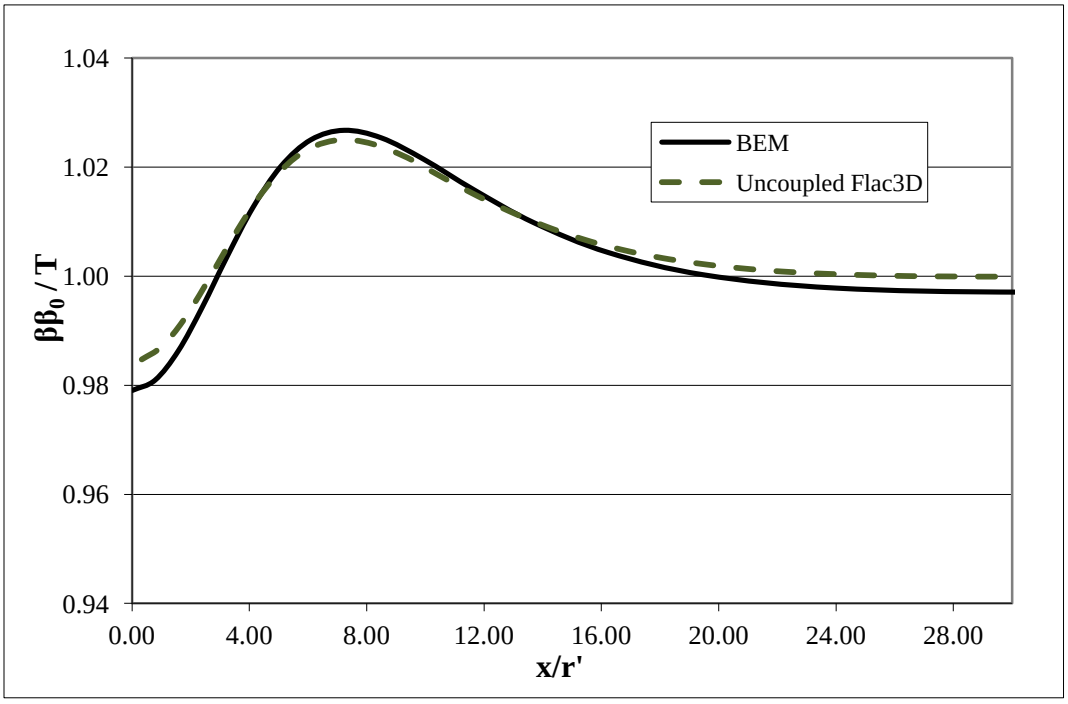


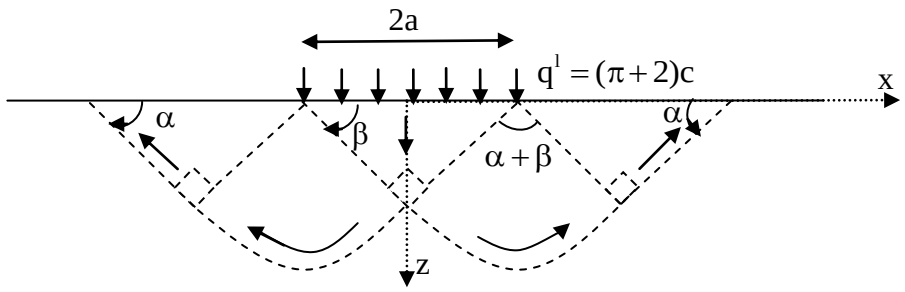
Figure 5.175 Uncoupled and exact normalized stress along the free traction straight edge of a semi-infinite plate *[d/r'= 10.0, α_1= 2.0]*

Table 5.47 BEM and coupled normalized stress along the free traction straight edge of a semi-infinite plate *[d/r'= 10.0, α_1= 2.0]*

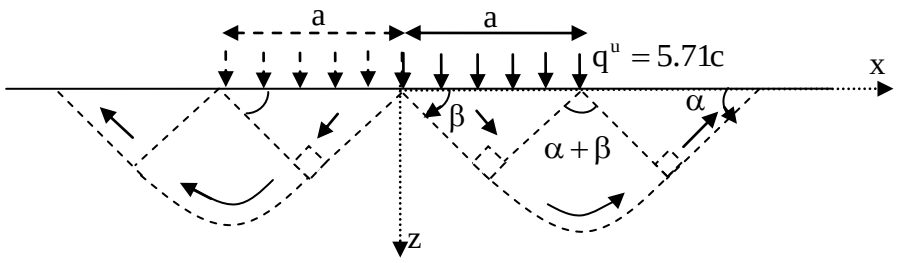
x/r'	BEM $\beta\beta_0/T$	First Coupled Method $\beta\beta_0/T$	R.D. %	Second Coupled Method $\beta\beta_0/T$	R.D %
0.0000	0.9791	0.9776	-0.148	0.9752	-0.3913
0.7831	0.9810	0.9794	-0.162	0.9774	-0.3661
1.5759	0.9863	0.9843	-0.202	0.9834	-0.2954
2.3888	0.9943	0.9917	-0.266	0.9924	-0.1943
3.2329	1.0036	1.0001	-0.345	1.0027	-0.0852
4.1214	1.0127	1.0082	-0.435	1.0127	0.0059
5.0697	1.0202	1.0148	-0.525	1.0207	0.0583
6.0973	1.0250	1.0188	-0.606	1.0256	0.0568
7.2290	1.0268	1.0200	-0.663	1.0268	0.0041
8.4980	1.0254	1.0184	-0.680	1.0245	-0.0819
9.9499	1.0214	1.0149	-0.642	1.0197	-0.1695
11.6498	1.0159	1.0103	-0.544	1.0136	-0.2258
13.6948	1.0098	1.0058	-0.395	1.0075	-0.2303
16.2367	1.0044	1.0021	-0.219	1.0025	-0.1820
19.5277	1.0002	0.9997	-0.057	0.9992	-0.1036
24.0211	0.9978	0.9984	0.060	0.9976	-0.0252
30.6226	0.9971	0.9982	0.113	0.9974	0.0265
41.4442	0.9976	0.9986	0.105	0.9980	0.0416
62.8210	0.9987	0.9993	0.062	0.9989	0.0287

5.3.5 Smooth Square Footing on a Cohesive Frictionless Material [134]

The analytical solution for the average footing pressure at failure of a smooth square footing on a cohesive frictionless elasto-plastic material (Tresca model) is not exactly obtained. However, Chen [135] presented two limits for the bearing capacity of the footing, see equations (5.33). The first limit is the upper bound (q^u) obtained using the failure mechanism of Shield and Drucker and based on a modified Hill failure mechanism, see Figure 5.176.b, and the second limit is the lower bound (q^l) which corresponds to the bearing capacity of a strip footing, obtained using Prandtl's failure mechanism, see Figure 5.176.a.



a. Prandtl failure mechanism



b. Modified Hill failure mechanism

Figure 5.176 Failure mechanisms under square footing on a cohesive frictionless material

$$q^u = 5.71c$$
$$q^l = (\pi + 2)c \qquad (5.33)$$

Where: c is the material (soil) cohesion, 2a is the footing length. α and β equal to 45^o for Prandtl's failure mechanism. α and β equal to 47^o 4' and 34^o for Shield et al. failure mechanism, respectively.

Although the size of the yield zone under the footing is in the range between (2a) and (3a), measured from the footing center, in the horizontal (x) direction for both mechanisms, no clear approximation is specified in the vertical (z) direction. This problem is solved in Flac^{3D} Manual as a verification problem. Only the bearing capacity of the footing is obtained numerically in this example, but the size of the yield zone is not discussed. The extension of this zone is studied here comparing its size in the coupled solution with the uncoupled one.

5.3.5.1 Uncoupled Flac^{3D} Solution

The used elasto-plastic Tresca material in this example has the same given material properties as in the Flac^{3D} verification example: Shear modulus (G) equals to 0.1 GPa; bulk modulus (K) equals to 0.2 GPa; cohesion (c) equals to 0.1 MPa; friction angle (φ) equals to 0^o; and dilation angle (ψ) equals to 0^o. The square footing half length (a) equals to 3.5 m. The Flac^{3D} bounded sub-domain model used to solve this problem is of radius L or ratio (L/a). This ratio is increased from the value 2.3 up to the value 5.1 and given the values 2.6, 3.4 and 4.3 between these two limits. Because of the problem's symmetry about xz and yz planes, only (1/4) of the loaded semi-sphere is modeled (see Figure 5.177). The Flac^{3D} bounded sub-domain's truncation boundary is fixed in all directions. Similar to the Flac^{3D} verification example, axes x and y are in the horizontal plane (the footing plane), and z axis is in the vertical direction (the applied velocity or load direction). Because the footing slab is assumed to be smooth, the nodes within a 3 m$\times$3 m of the loaded area (in plane z = 0) are free in x and y directions; and a velocity of magnitude 2.5 $\times 10^{-5}$ m/step is applied in z direction at these nodes (total steps number is 8000 steps). The bearing area, for the case of applied velocity loading, is assumed in this example to extend, measured from the footing center, to half the distance between the last applied node and the next node, thus (a) equals to 3.5 m.

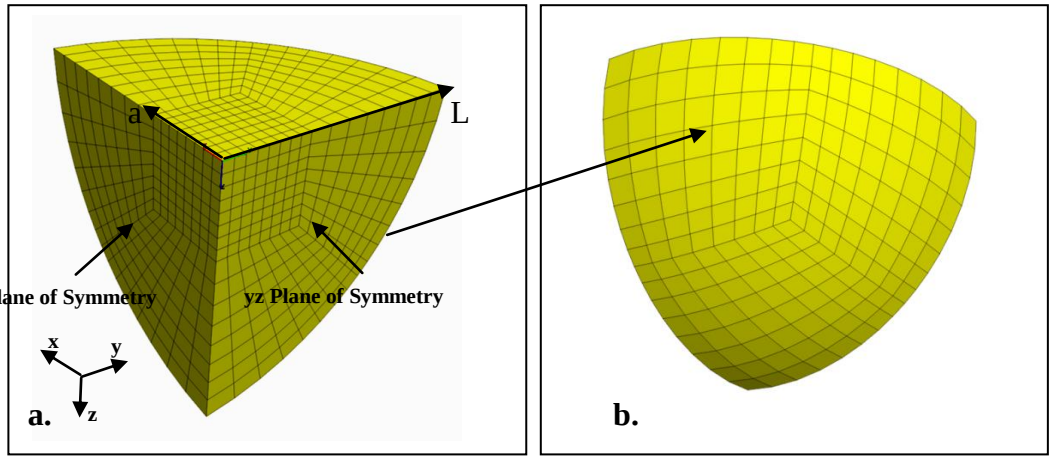


Figure 5.177 a. Flac3D Model for the square footing on a cohesive frictionless material problem
b. Flac3D-BEM sub-domains' interface

The uncoupled Flac^{3D} solutions for the normalized average footing pressure (q/c) using the mentioned Flac^{3D} model of five different ratios (L/a) and the corresponding relative difference (RD) with the analytical normalized lower bound value (q^l/c) for the footing bearing capacity are computed and listed in Table 5.48. It is clearly observed that the computed numerical solutions of (q/c) agree with the analytical lower bound value (less than 1% RD) disregard the size of Flac^{3D} bounded sub-domain used to obtain these solutions (see Table 5.48). The normalized average footing pressure (q/c), given the name p_load, is plotted against the normalized vertical displacement (u_z/a) at the center of the footing, given the name c_disp, along with the upper bound, given the name p_solup, and the lower bound, given the name p_sollo, for the footing bearing capacity (see Figure 5.178). The figure shows the agreement between the numerical and the analytical lower bound values of the footing bearing capacity.

Table 5.48 the uncoupled Flac3D solutions for the footing normalized bearing capacity and the size of yield zone

			Normalized Bearing Capacity				
			Size of Yield Zone		Analytical Solutions		Numerical Solution [q/c]	RD % = $(q-q^l)/q^l$
L/a	No. of Zones	CPU (seconds)	Lx/a	Lz/a	Upper Bound [q^u/c]	Lower Bound [q^l/c]		
2.3	864	17.7	2.20	1.90	5.710	5.142	5.145	0.06
2.6	2048	34.2	2.30	2.00			5.160	0.35
3.4	6912	103.4	2.30	2.25			5.140	-0.03
4.3	16384	222.7	2.30	2.50			5.135	-0.14
5.1	32000	444.4	2.30	2.50			5.132	-0.20

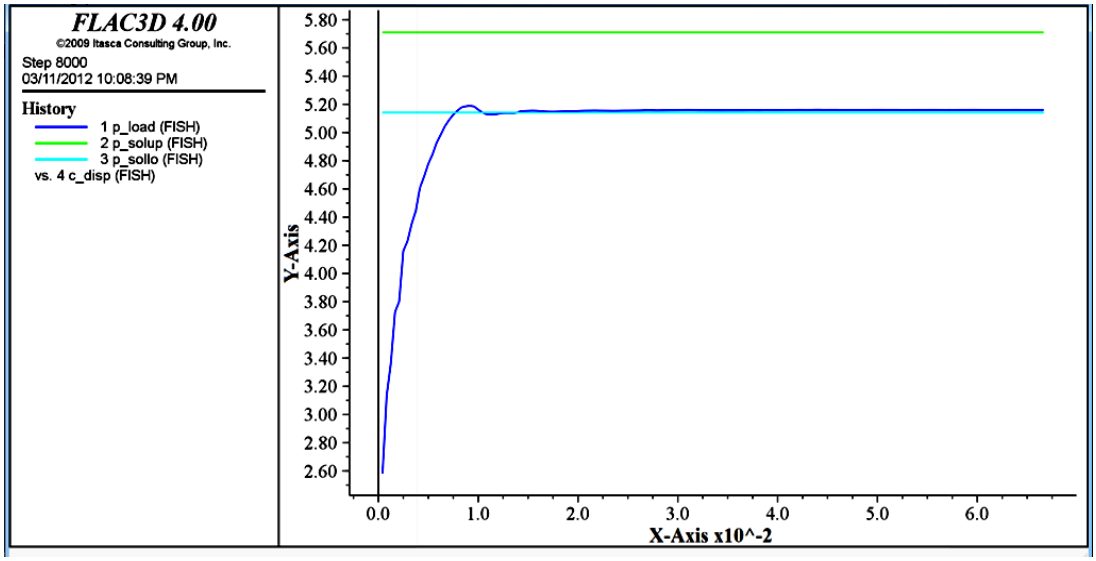


Figure 5.178 The uncoupled Flac3D solution for the normalized footing pressure (q/c) versus the normalized vertical displacement (u_z/a) at the center of the footing, *(Y-Axis ≡ q/c) and (X-Axis ≡* u_z/a*).*

To evaluate the size of the Yield or Plasticity zone the inverse of the factor of safety is derived, according to section 3.6.4.2 (Mohr-Coulomb Model), utilizing Mohr-Coulomb failure criterion in equation (3.90) as the following:

$$\text{Resisting shear stress} = 2c\sqrt{N_\phi} - \sigma_1(N_\phi - 1) \tag{5.34}$$

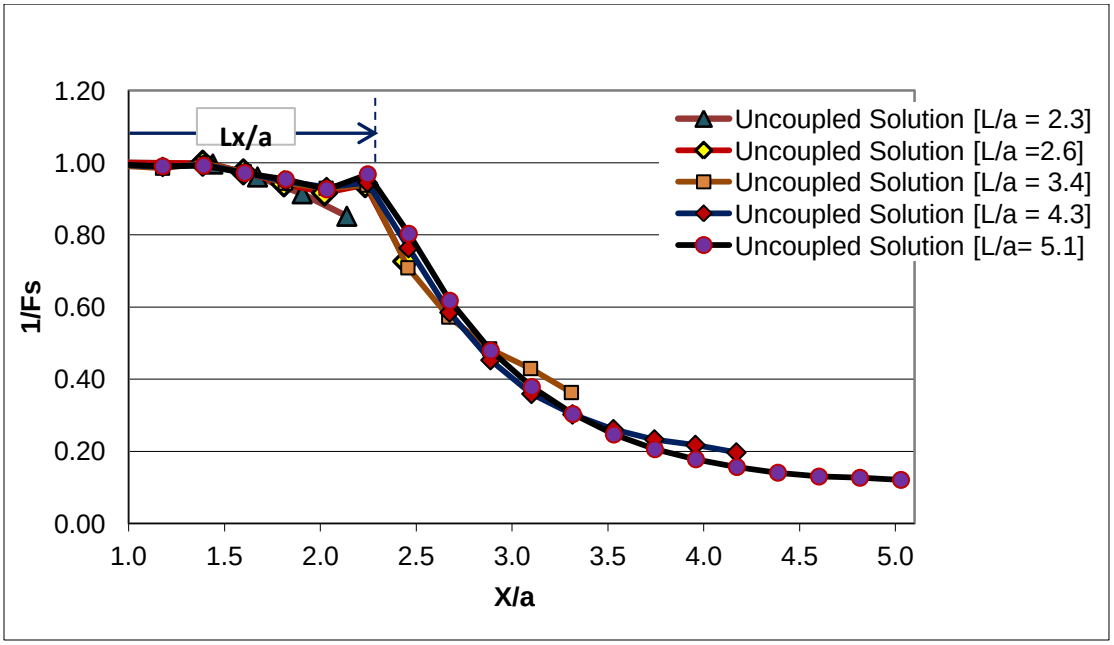
$$\text{Acting shear stress} = \sigma_3 - \sigma_1 \tag{5.35}$$

$$\text{Inverse Factor of Safety} = \frac{(\sigma_3 - \sigma_1)}{2c\sqrt{N_\phi} - \sigma_1(N_\phi - 1)} \tag{5.36}$$

It should be mentioned that because in this example ϕ equals to 0, constant N_ϕ equals 1 and Mohr-Coulomb criterion transforms into Tresca criterion (see equation 3.91). The principal stresses are arranged as: $\sigma_1 \leq \sigma_2 \leq \sigma_3$.

The inverse of the factor of safety (1/Fs) is computed in the zones along x axis (in plane z = 0) to estimate the size of the Yield zone inside Flac^{3D} sub-domain at the ground level, and in the zones along z axis to estimate the size of the Yield zone in the same sub-domain in the vertical direction, using a FISH function -safetyfactor- created by the author. The normalized length, using the uncoupled Flac^{3D} method, of the Yield zone (Lx/a = 2.3) in x direction converges into constant solution starts at Flac^{3D} model of ratio (L/a = 2.6), but the normalized length (Lz/a = 2.5) in the vertical z direction does not converge until the size of Flac^{3D} sub-domain increased to become of ratio (L/a = 4.3) (see Table 5.48 and Figure 5.179).

a.



b.

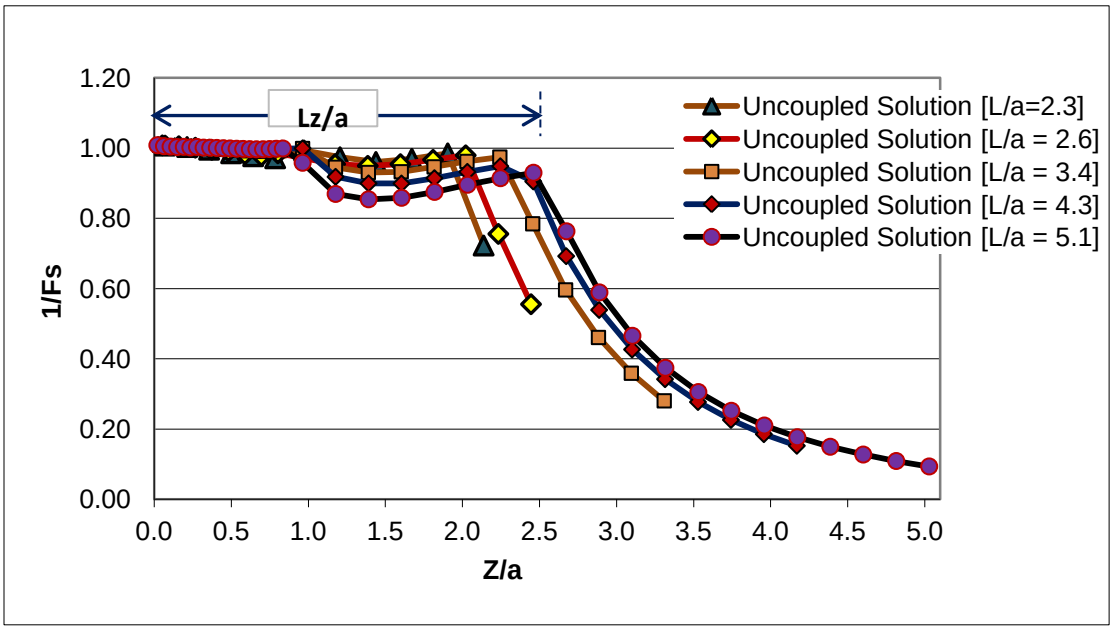


Figure 5.179 Uncoupled Flac3D solution for inverse of the factor of safety and the size of Yield zone in: a. x and b. z directions.

5.3.5.1 Coupled Flac^{3D}-BEM Solution (First Method/IDDM)

The coupled solution for the normalized average footing pressure (q/c) using the same Flac^{3D} model and by applying the first method (IDDM) did not differ from the uncoupled Flac^{3D} solution (RD is less than 1% in both coupled and uncoupled solutions using different ratios L/a), see Table 5.49 and Figure 5.180.

Table 5.49 Coupled and uncoupled solutions for the footing normalized bearing capacity, the size of yield zone and the required runtime (CPU).

Uncoupled Flac3D Solution [L/a = 5.1]							
Normalized Bearing Capacity							
Exact Solution		**Numerical Solution [q/c]**	**R.D.% = $(q-q^l)/q^l$**	**Yield Zone [Lz/a]**	**CPU [Second]**		**Number of zones**
Upper Bound [q^u/c]	**Lower Bound [q^l/c]**						
5.710	5.142	5.132	-0.19	2.50	444.4		32000
Coupled Solution [L/a = 2.6]							
Normalized Bearing Capacity							
Exact Solution		**Numerical Solution [q/c]**	**R.D.% = $(q-q^l)/q^l$**	**Yield Zone [Lz/a]**	**CPU [Second]**		**Number of zones**
Upper Bound [q^u/c]	**Lower Bound [q^l/c]**				**Pre-processing**	**Post-processing**	
5.710	5.142	5.162	0.39	2.50	360.10	10.5	2048

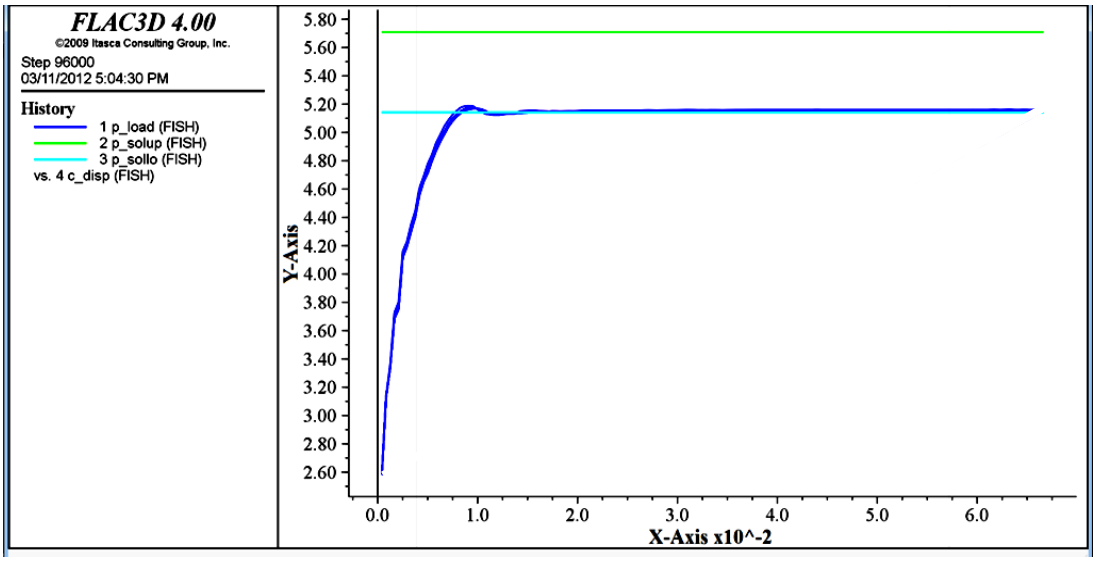
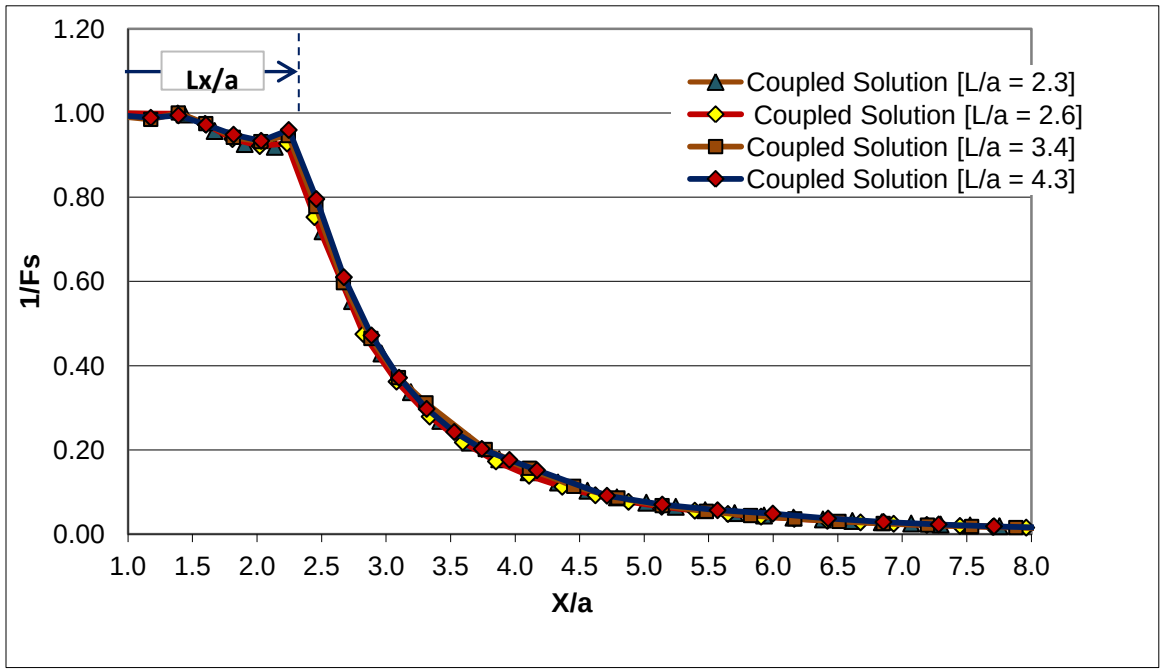


Figure 5.180 Coupled Flac3D-BEM solution for (q/c) versus (u_z/a), *(Y-Axis ≡ q/c) and (X-Axis ≡ u_z/a).*

The inverse of the factor of safety (1/Fs) in the coupled solution is computed in the zones along x and z axes to estimate the size of the Yield zone inside Flac^{3D} sub-domain at the ground level, and in the vertical direction, respectively using the FISH function (safetyfactor). FISH functions called Postst and COLOUMB1/COLOUMB2 are used to Compute (1/Fs) in the BEM semi-infinite sub-domain along the same axes, and to estimate the size of the Yield zone in the same directions beyond the truncation boundary limits. The Flac^{3D} bounded sub-domain ratio (L/a) is given the values 2.3, 2.6, 3.4 and 4.3, respectively. Contrary to the uncoupled solution, the obtained size of the Yield zone in the coupled solution at the ground level (Lx/a = 2.3) and in the vertical direction (Lz/a = 2.5) was the same regardless of the size of the used Flac^{3D} bounded sub-domain model, (see Figure 5.181.a, and Figure 5.181.b). The required, runtime (CPU) in the coupled method is 10% less than the time in the uncoupled method to obtain the converging solution (see Table 5.49).

a.



b.

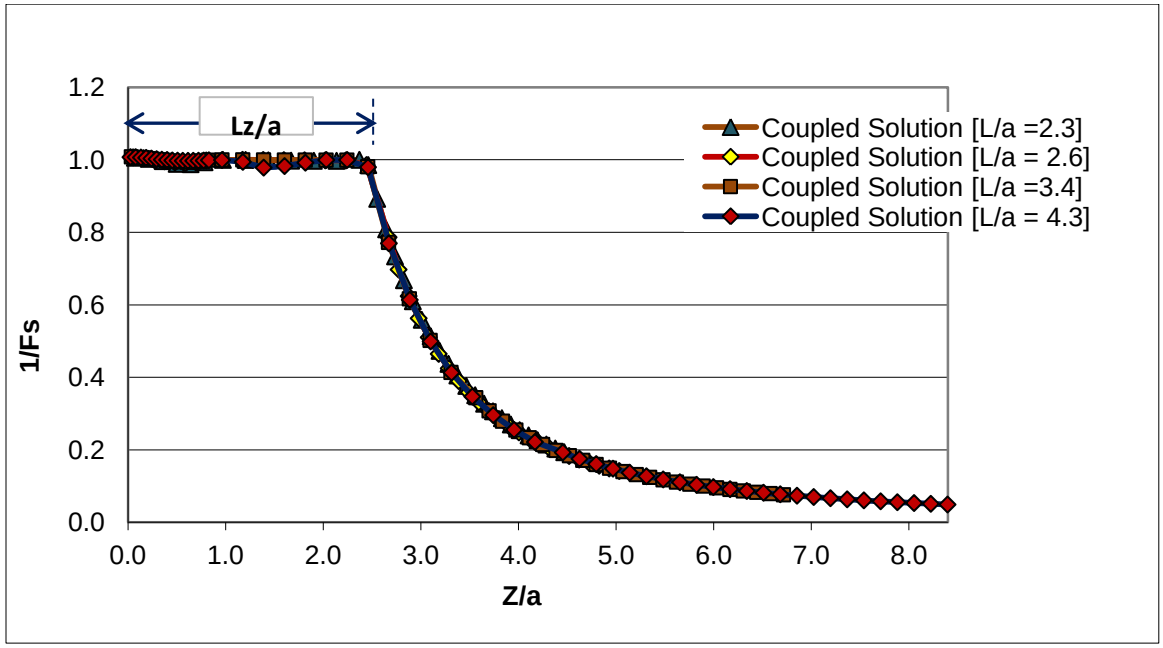
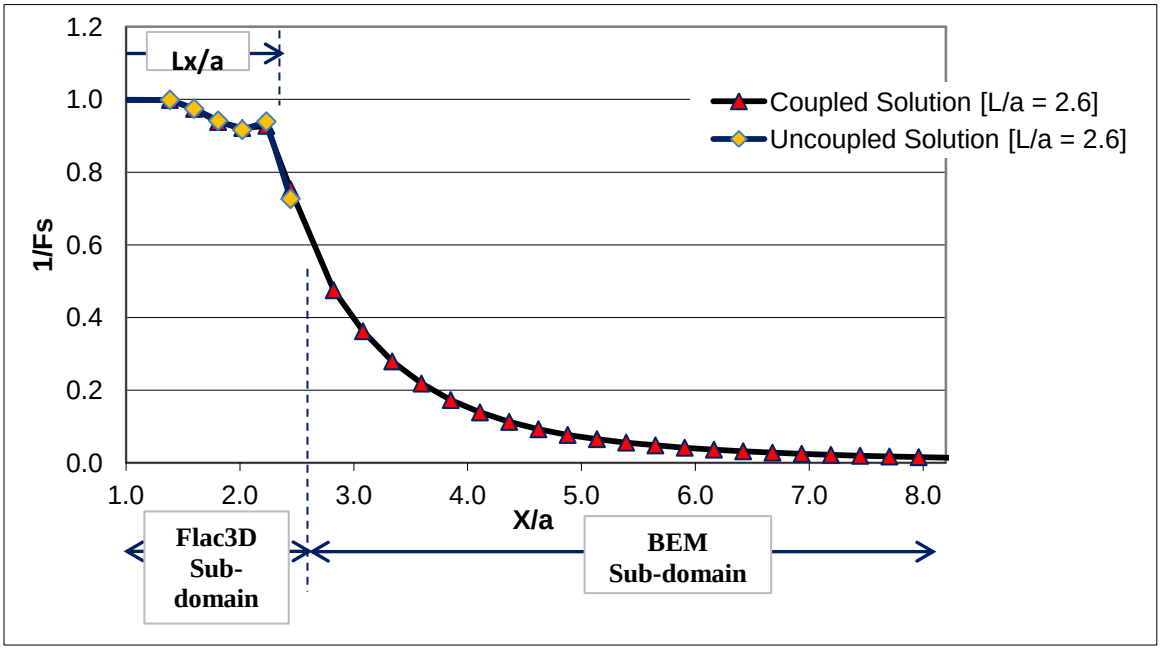


Figure 5.181 Coupled Flac3D-BEM solution (First Method/IDDM) for the inverse of factor of safety and the size of Yield zone in: a. x and b. z directions.

A comparison between the Yield zone size that corresponds to the coupled and uncoupled solution is illustrated in Figure 5.182.a (x direction) and in Figure 182.b (z direction). Both

solutions are obtained using the same Flac^{3D} sub-domain size (L/a = 2.6). Although, the Yield zone has the same size at the ground level in both solutions, its size in the vertical direction is 20% less in the uncoupled solution than the coupled one.

a.



b.

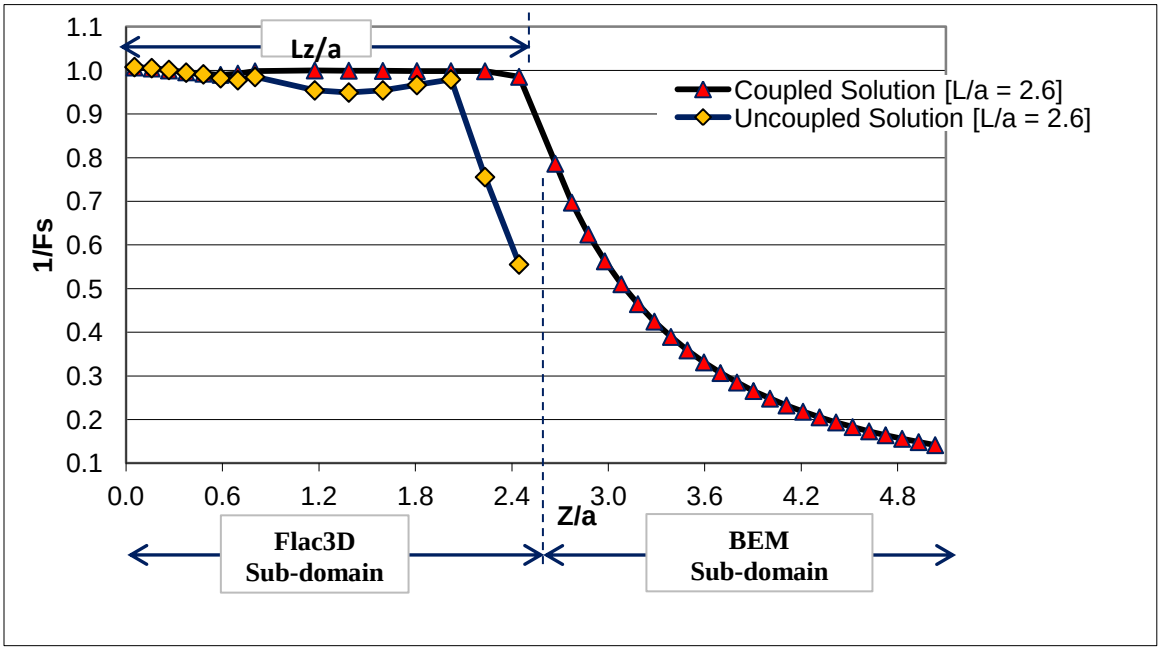
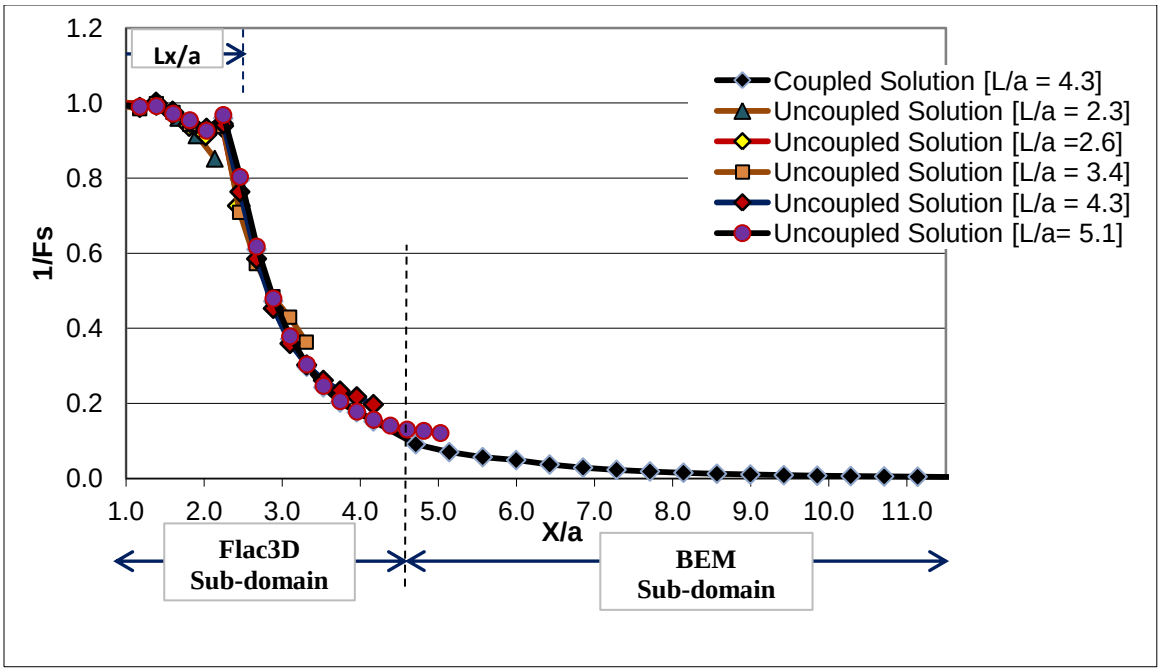


Figure 5.182, a. & b. Coupled and uncoupled solutions for the inverse of the factor of safety and the size of Yield zone in: a. x and b. z directions, L/a = 2.6.

The uncoupled solution converges into the coupled solution by increasing the Flac^{3D} size (increasing L/a) as illustrated in Figures 5.183.a (x direction) and 5.183.b (z direction).

a.



b.

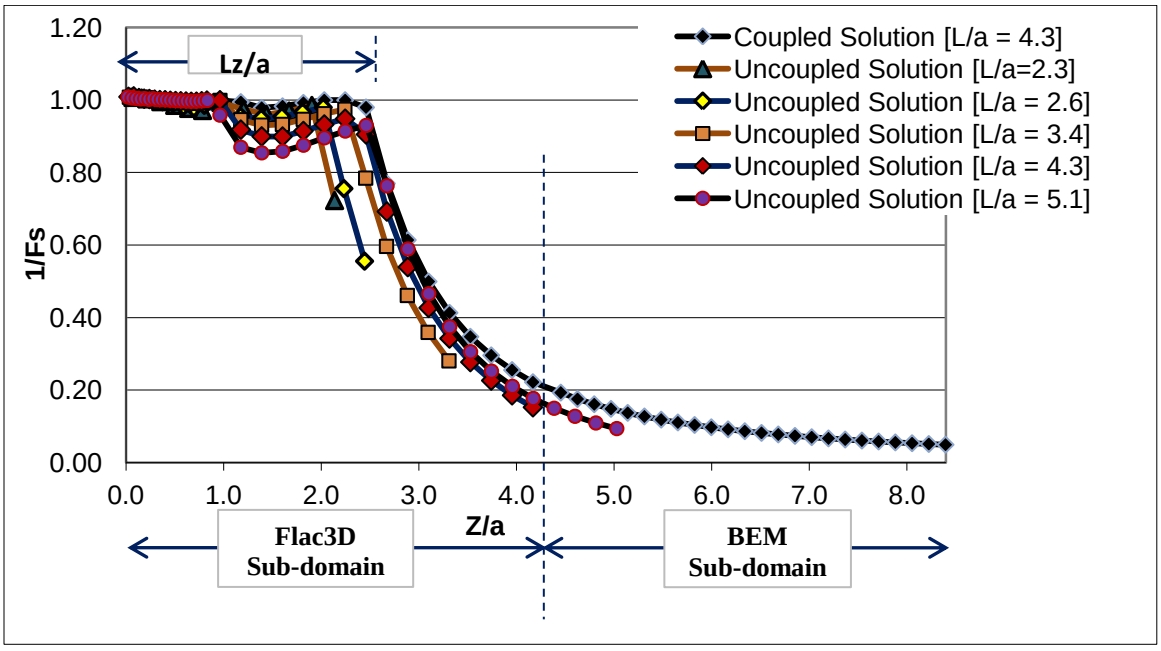


Figure 5.183 Coupled and uncoupled solutions for the inverse of the factor of safety and the size of Yield zone in: a. x and b. z directions, L/a: 2.6-5.1.

The Yield zone in the coupled and uncoupled solutions is also visualized in three dimensional graphs as shown in Figure 5.184 in Appendix e, p.339. The velocity contour and velocity vectors and maximum shear contour in the coupled solution are also illustrated in Figures 5.185.a and 5.185.b in Appendix e, p.340, respectively.

5.4 Flac^{3D} Commands and Fish Functions Designed to Compute the Uncoupled Solutions

The written codes by the author to solve each example are composed of two main parts: the first and the second parts are created to obtain the solution to the uncoupled and coupled problems respectively. The first part is created to compare both solutions while computing the required total number of steps to achieve equilibrium using Flac^{3D} independently. The total number of steps required for the coupled solution, are computed only for the stress boundary problems, not for the displacement boundary ones, where the step number is decided a priori. The Flac^{3D} program performs its own computations over this number of steps as seen in the coupling algorithm (in section 5.2) at each iteration until the targeted convergence is reached. The following is a brief explanation of the Flac^{3D} commands and Fish codes created by the author to compute the uncoupled Flac^{3D} solutions for:

5.4.1 The Spherical Cavity in 3D Infinite Medium Problem

The geometry of the Flac^{3D} model to solve the spherical excavation problem is generated using a Flac command (*gen zone radbrick*) and a Fish function given the name (*make_sphere*). This function is quoted from the Flac^{3D} manual [129] after minor modifications made by the author to generate a spherical outer boundary in the model. A field stress is initialized in the model (command: *initial*) and z stress is applied over the truncation boundary (command: *apply*). The planes of symmetry are fixed using the commands: *apply xv 0 range x -0.01 0.01*, etc. The total number of steps is computed by two Fish functions called (*totalstepnumber1* and

totalstepnumber). *nastr* is a Fish function created to obtain Flac^{3D} and analytical z stress solutions z in plane (z=0). *Nadis* is another Fish function which find the x displacement along the X axis and z displacement at point A; both are computed by Flac^{3D} (see Appendix g, pp.373-377).

5.4.2 The Square Uniform Load on 3D Semi-infinite Medium Problem

The planes of symmetry are fixed using the commands: *apply xv 0 range x -0.0001 0.0001 and apply yv 0 range y -0.001 0.001*. On the other hand, the spherical truncation boundary model is fixed by a Fish function named *fixboundary* in z direction only. The uniform load is applied on a square area of the surface using command: *apply szz @Load range x 0 ,@XLength y 0, @YLength z 0.001 -0.001*, and a zero field stress is initialized in the model by the command: *ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress*. *nastr* is a Fish function created to obtain Flac^{3D} and analytical z, x and xz stress solutions under the corner of the loaded area, whereas *nastr1* function finds z and x stress solutions under the center of the load. *Nadis* function acquires the z displacement (numerical and analytical) at the grid points directly located under the load corner and at point B on the surface. Whereas, *nadis1* function obtains the z displacement at grid points along z axis under the load center and at point A (see Appendix g, pp.378-387).

5.4.3 The Cylindrical Tunnel in 2D Infinite Medium Problem

The planes of symmetry are fixed using the commands: *apply xv 0 range x -0.001 0.001 z @red @len, apply xv 0 range x -0.001 0.001 z @radf @lenf, apply zv 0 range z -0.001 0.001 x @rad @len.* Because it is a plane strain problem, the model nodes are fixed in y direction (the tunnel axis) by the command: *apply yv 0.* The field stress is initialized equally in both x and z directions and the same initial stress components are applied to the truncation boundaries with:

apply and apply remove commands. *Nastr* is a Fish function that obtains solutions for Flac^{3D} and analytical normalized radial and tangential stresses at the zone centroids close to the X axis, while *nadis* function finds the numerical and analytical values of the radial displacement at the grid points along the X axis (see Appendix g, pp.388-392). This problem solution is based on Flac^{3D} manual solution of Salencon problem [130].

5.4.4 The Hole near the Edge of a 2D Semi-infinite Plate under Tension Problem

Two model configurations are built to solve this problem. The first model is a hollow cylinder generated by the command *gen zone cshe.* Because this configuration does not represent the straight edge of the plate, the stress along this edge cannot be computed in the uncoupled phase of the solution (see Appendix g, pp.392-394). Conversely, the second configuration created using two commands (*gen zone radcyl* and *gen zone radtunnel*), represents the plate's straight edge and the stress (BEM and Flac^{3D} solutions) along the edge is found by the Fish function *nastr*. The geometry of the problem is generated with two primitives. The second *radtunnel* primitive is added to the first one to approximate the unbounded extensions of the semi-infinite plate in x and z directions (see Appendix g, pp.394-397).

5.4.5 Smooth Square Footing on a Cohesive Frictionless Material

safetyfactor is a FISH function created to obtain the inverse of the factor of safety (1/Fs) in the zones along x and y axes (in plane z = 0) to estimate the size of the Yield zone inside the Flac^{3D} sub-domain at the ground level, and in the zones along z axis to estimate the size of the Yield zone in this sub-domain in the vertical direction, see Appendix g, pp.398-404. Another FISH functions *Poststt* and *COLOUMB1/COLOUMB2* are created to Compute (1/Fs) in the BEM semi-infinite sub-domain along the same axes, and to estimate the size of the Yield zone in the same directions beyond the truncation boundary limits, see Appendix g, pp.490-496.

5.5 Coupled Solution Code

The main structure of the code is illustrated in the flow chart, shown in Figure 5.186. It follows the uncoupled solution code in the first four examples. The code that solves the spherical cavity in 3D infinite medium problem is displayed completely in Appendix g, pp.407-431. Because most of the created Fish functions of the code are the same in the four examples, only the modified functions, numbered from 1 to 8 (see Figure 5.186) are displayed for the other three examples (see Appendix g, pp.432-462). The modification depends mainly on the geometry of the different Flac^{3D} primitive, generated for each example. These functions should be placed in the same order shown in Figure 5.186 and in Appendix g, pp. 407-431. The code that solves the smooth square footing on a cohesive frictionless material problem in a 3D semi-infinite medium is displayed completely in Appendix g, pp.463-496.

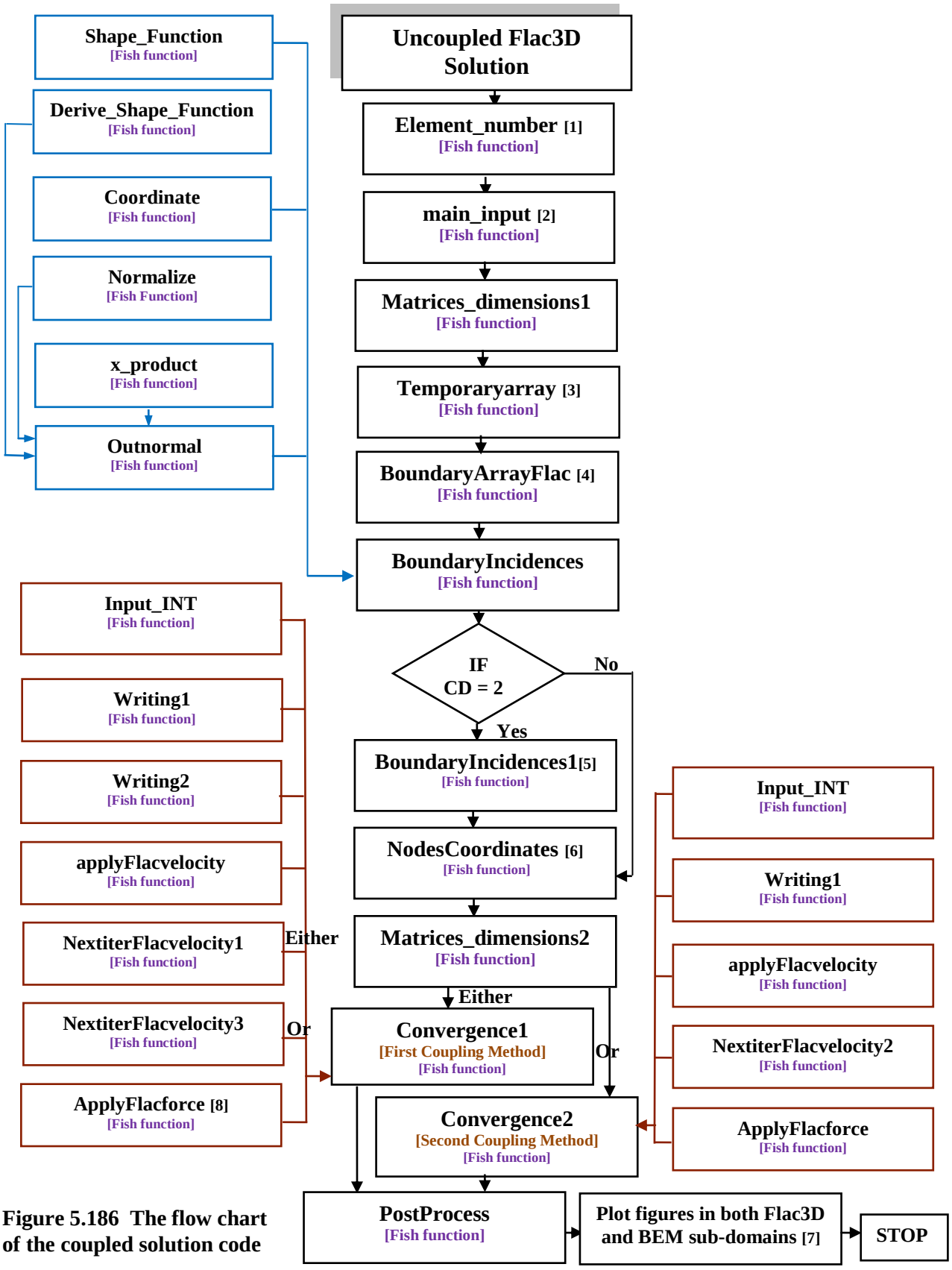


Figure 5.186 The flow chart of the coupled solution code

5.6 The Flow Chart of the Coupled Solution Code

The Fish functions, seen in Figure 5.186, are created to accomplish the following objectives:

Element_number: obtains the boundary elements' total number of the interface, discretized by the Flac^{3D} program. Flac^{3D} discretizes the primitive into a number of zones according to the user's input. The base of the zones adjacent to the interface is counted as boundary elements by this function according to a condition dependent on the geometry of the used Flac^{3D} primitive.

Main_input: assigns values to the problem's fundamental parameters (such as the problem of Cartesian dimensions, number of the degree of freedom, the symmetry code, type of sub-domain and other parameters)

Matrices_dimensions1: declares some of the used arrays in this code and defines their dimensions.

Temporaryarray: obtains a boundary elements temporary incidences array of the interface called IncdTemp. The incidences numerated by Flac^{3D} are assembled according to a condition dependent on the geometry of the used Flac^{3D} primitive. The maximum boundary nodes number is also obtained by this function from the Flac^{3D} program.

BoundaryArrayFlac: creates two temporary arrays of the interface BEs incidences to be transformed into a BEM incidence array later in this code. This function also obtains a temporary array of the boundary nodes coordinates, extracted from Flac^{3D} program.

BoundaryIncidences: obtains the outward normal vector at the center of each boundary element (the intrinsic coordinates: ξ and η equal to 0) by calling for another Fish function named **Outnormal** which in turn calls for the following Fish functions:

- **Shape_Function:** computes a first order shape function vector for one and two-dimension boundary elements at given intrinsic coordinates.
- **Derive_Shape_Function:** computes the derivative array of the first order boundary element shape function vector at given intrinsic coordinates.
- **Coordinate:** obtains the Cartesian coordinates for a point with given intrinsic coordinates (shape function).
- **Normalize:** normalize a given vector.
- **x_product:** computes the cross product of two given vectors. The computed vector is normal to the two given vectors.

Function **BoundaryIncidences** also rearranges the BEs' incidences in the temporary arrays in a way that guarantees solving an external problem by directing the outward normal vector away from the unbounded body of the BEM sub-domain. Afterward, this function renumbers the incidences to become usable by the BEM code, and computes the total boundary nodes. **BoundaryIncidences1** is very similar to Function BoundaryIncidences, but it is designed for solving 2-D problems only.

NodesCoordinates rearranges the temporary array of the boundary nodes' coordinates, extracted from the Flac^{3D} program to become usable by the BEM code.

Matrices_dimensions2: declares another group of arrays and defines their dimensions.

Convergence1: performs the first coupling method, described before in section 5.2, over an iteration loop. The following Fish functions are recalled by this function:

- **Input_INT:** reads the internal point coordinates and writes the boundary nodes' coordinates and incidences.

- **Writing1** and **writing2:** write the interface boundary elements' displacement and traction arrays, the error and the renewed displacement array at a chosen iteration.
- **NextiterFlacvelocity1** computes the velocity for the next iteration (i+1), according to the relaxation equation 4.83, to be applied over the Flac^{3D} sub-domain side of the interface by the Fish function named **applyFlacvelocity.** The iteration stops whenever the convergence condition of equation 5.32 is satisfied. If **NextiterFlacvelocity3** is recalled by function convergence1, the same relaxation equation is used but with a different convergence condition defined before in equation (4.82).

This function obtains the interface boundary elements' displacement array (the interface is approached from the BEM sub-domain) at iteration i using a C++ intrinsic, created and named by the author as **example_PreProcessing.** The BEM code, which performs the numerical analysis over the unbounded BEM sub-domain, is written in C++ programming language, compiled as a DLL file (dynamic link library) and given the name *BEMKMM002_64.dll*. Whenever, Flac^{3D} loads the DLL file, using the command *load function*, **example_PreProcessing** behaves very similar to any of the predefined Fish intrinsics (such as atan, cross, ln, z_id, etc.) [131]. C++ Fish intrinsic plug-ins has two main advantages: The first is "The *FISH* intrinsic uses a C++ interface that provides access to the internal structure of *FISH*, as well as the data of *FLAC^{3D}* " [131]. Therefore, the input and output of both programs are exchanged between the Flac^{3D} program and the BEM code internally without losing any of the solution accuracy. The second advantage is "C++ intrinsics should be from 10 to 100 times faster to execute than *FISH* functions" [131]. This is confirmed, in practice, by transforming the BEM code (pre and post processing) completely into *FISH* language code. This code, used

to solve the spherical excavation problem, was extremely slower than the created C++ intrinsics.

C++ Fish intrinsic example_PreProcessing takes 17 arguments. The first 15 arguments are a given input, and the rest (BELU and BELT) are two arrays computed and returned by the intrinsic. BELU(NELM,DOFL) and BELT(NELM,DOFL) are the interface boundary elements displacement and traction arrays, respectively. E**xample_PreProcessing** transforms the interface nodal forces vector, obtained by the Fish function **ApplyFlacforce** from $Flac^{3D}$ program, into traction; then it computes the interface BEs nodal displacement array at every iteration.

Convergence2: this function is an alternative to function Convergence1. It performs the second coupling (SDDM) method, explained in section 5.3.1.4, over an iteration loop. Convergence2 recalls almost the same Fish functions used by Convergence1 before, such as: Input_INT, writing1, applyFlacvelocity and ApplyFlacforce. However, the velocity for the next iteration (i+1), is computed by function **NextiterFlacvelocity2** according to equation 5.4, and applied over the $Flac^{3D}$ side of the interface by **applyFlacvelocity.**

PostProcess: computes the interface BEs stress components and the displacement and the stress components at chosen internal nodes (inside the BEM unbounded sub-domain). Another intrinsic C++ plug-ins function is created and named also by the author as **example_PostProcessing**, which performs these computations as a post processing step of the BE numerical analysis method. The BEM post processing C++ code is compiled in the same DLL file *BEMKMM002_64.dll*. **example_PostProcessing** takes 23 arguments. The first 20 arguments are a given input, which includes the earlier computed interface BEs' displacement

and traction arrays using onvergence1 or Convergence2. The last three arguments are the computed boundary stress, internal displacement, and stress arrays, respectively.

See Variables, Parameters and Arrays Used in the Coupled Solution Codes in Appendix g, pp.405-406.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Two coupling methods were developed in this thesis. The first developed coupling method (IDDM), the Domain Decomposition Sequential Dirichlet-Neumann Iterative Coupling method, has the following advantages:

1. There is no need in this iterative method to combine the matrices of the BEM sub-domain with the FDM/FEM sub-domain (the Flac^{3D} program, which solves non-linear problems as fast as the linear ones, does not store global stiffness matrices). In contrast, coupling methods at the level of discretized equations require building a complicated asymmetric unified system of equations.
2. Because the Iterative method allows the FDM (Flac^{3D} program) and the BEM program to work independently and interactively, no fundamental modifications of the existing BEM and FDM/FEM computer codes are required.
3. The method eliminated the error introduced by the truncation boundary, corrected the mechanical responses in the Flac^{3D} bounded sub-domain, and reduced the computer runtime (CPU) by reducing the required zone numbers in the uncoupled Flac^{3D} method.
4. The mechanical responses very far from the Flac^{3D} bounded sub-domain are computed with less cost in runtime or in the required number of finite difference constant strain rate zones.
5. The method achieved full solution convergence and complete stress and displacement continuity across the interface. It has proved to be independent of mesh size, iterations number, truncation boundary location and material properties.

6. The coupled solution accuracy in the four 3D and 2D infinite and semi-infinite applications reached by this method, in comparison with the analytical solution, is very high (less than 1% R.E.). This accuracy, especially in the 3D semi-infinite application, is barely obtained by the uncoupled Flac^{3D} method.
7. The saved runtime (CPU) in the coupled method is higher in the three dimensional than the two dimensional problems, and in the semi-infinite than infinite domains. The CPU required to reach the coupled solution in 3D semi-infinite problems was 1/2 the required time to reach the uncoupled Flac^{3D} solution.
8. Because this method is truncation boundary location independent, the analysis time needed to accomplish the ideal TBL in the uncoupled Flac^{3D} method is completely saved.
9. The derived complementary part of the Kernels $\mathbf{S}^{c}$ and $\mathbf{R}^{c}$, based on Mindlin's fundamental solutions and required to compute the stress inside the 3D semi-infinite domain, is tested and proved to be accurate in this thesis.
10. The two BEM pre and post processing C++ *FISH* intrinsic plug-ins created in this thesis are also demonstrated to be accurate and faster to execute than *FISH* functions.
11. The minimum number of iterations (NIT) needed to reduce the initial error by a factor η is observed to be independent of the number of zones (number of degrees of freedom).
12. The first method (IDDM) succeeded in coupling the elastic/linear and plastic/non-linear behavior of Flac^{3D} bounded sub-domain with the linear/elastic behavior of the surrounding BEM infinite or semi-infinite sub-domain.
13. The size of Yield/Plastic zone was determined by the first coupling method/IDDM with higher certainty and with less runtime (CPU) than the uncoupled Flac^{3D} method.

Although the second coupling method, the Single Step Domain Decomposition Sequential Dirichlet-Neumann (SDDM), is faster than the first coupling method, it has the following disadvantages:

1. A less accurate solution is produced by this method, compared to the first coupling method.
2. The second method does not reach a full solution convergence and does not develop the stress and displacement continuity across the interface.
3. Similar to the uncoupled Flac^{3D} solution, the second method solution is truncation boundary position dependent.
4. Even if the second method solution is not affected by Young modulus material property value, it is dependent on Poisson's ratio υ values. The higher Poisson's ratio is, the farther the truncation boundary should be positioned.
5. The method is applicable practically only for low Poisson's ratio υ values (0.0-0.2).

6.2 Recommendations

1. The developed coupling methods in this thesis are applied over an isotropic material using fundamental solutions in the BIE derived for this type of material. The same methods can be applied to anisotropic material using the fundamental solutions derived by Tonon et al. [132].
2. In general, because the medium in Geomechanics is multi-layered, every layer has its own material properties, the multiple regions BEM approach [2], [3], [133] should be utilized to solve the problems that exist in this medium. The interfaces between these layers, which may extend infinitely or semi-infinitely, are considered as boundaries and discretized into isoparametric finite and infinite BEs (see sections 2.2.8.2 and 5.3.2.1).

3. The developed coupling methods in this work can be employed to couple the nonlinear elasto-plastic behaviour in both sub-domains: the bounded sub-domain analyzed by Flac^{3D} and the infinite or semi-infinite sub-domain analyzed by nonlinear BEM [3], [5], [43], [133].
4. Coupling Flac^{3D} program, which performs a dynamic analysis in a bounded sub-domain, with the BEM [3], [5], which performs the dynamic analysis in the surrounding unbounded sub-domain and satisfies implicitly the radiation conditions, can solve the problem of reflecting waves at the introduced truncation boundaries.
5. Improve the performance of the used BEM code to further reduce the runtime (CPU) and increase the efficiency of the developed coupling methods.
6. Although, coupling the Flac^{3D} program with BEM applications in this thesis were only with elastic and elasto-plastic (Mohr-Coulomb/Tresca) material constitutive models, more applications with time independent models, such as the Drucker-Prager model, or time dependent models, such as visco-elastic or visco-plastic constitutive models, could be developed in the future.

References

[1] R. V. Southwell, *Relaxation Methods in Theoretical Physics*. Oxford, UK: Oxford University Press, 1946.

[2] J. H. Kane, *Boundary Element Analysis in Engineering Continuum Mechanics*. Englewood Cliffs, New Jersey: Prentice Hall, 1994.

[3] G. Beer, I. Smith, and C. H. Duenser, *The Boundary Element Method with Programming for Engineers and Scientists*. Springer Wien, New York, 2008.

[4] R. D. Mindlin, "Force at a point in the interior of a semi-infinite solid," *Physics,* vol. 7, pp. 195-202, 1936.

[5] C. A. Brebbia, J. C. F. Telles, and L.C. Wrobel. *Boundary element Techniques, Theory and Applications in Engineering*. Springer-Verlag, New York, 1984.

[6] W. Moser, C. H. Duenser, and G. Beer, "Mapped infinite elements for three-dimensional multi-region boundary element analysis," *Int J Numer Meth Engng,* vol. 61, pp. 317-328, 2004.

[7] M. Guiggiani, and A. Gigante, "A general algorithm for multidimensional Cauchy principal value integrals in the boundary element method," *Journal of Applied Mechanics,* vol. 57, pp. 906-915, 1990.

[8] O. J. B. Almeida Pereira, and P. Parreira, "Direct evaluation of Cauchy-principal-value integrals in boundary elements for infinite and semi-infinite three-dimensional domains," *Engineering Analysis with Boundary Elements,* vol.13, pp.313-320, 1994.

[9] V. Mantic, "A new formula for the C-matrix in the Somigliana identity," *Journal of Elasticity,* vol. 33, pp.191-201, 1993.

[10] J. C. F. Telles, and C. A. Brebbia, "Boundary element solution for half-plane problems," *Int. J. Solids Structures,* vol., no. 12, pp. 1149-1158, 1980.

[11] W. Ritz, "Ubereine Methode zur Losunggewisser Variations-Probleme der Mathematischen Physik," *Journal* für *reine und angewandte Mathematik,* vol. 135, pp. 1-61, 1909.

[12] O. C. Zienkiewicz, *The Finite Element Method*. Maidenhead: McGraw-Hill, 1977.

[13] D. R. J. Owen, and E. Hinton, *Finite Elements in Plasticity: Theory and Practice.* Swansea, UK: Pineridge Press Limited, 1980.

[14] I. Fredholm, "Solution d'un problem fundamental de la theorie de l'elasticitie," *Arkiv Mat Astron. Fys,* vol. 2, pp. 108, 1905.

[15] S. G. Mikhlin, *Multi-Dimensional Singular Integrals and Integral Equations*. Oxford: Pregamon Press, 1965.

[16] E. Trefftz, "Ein Gegenstuckzum Ritz'schen Verfahren," Proceedings 2nd International Congress in Applied Mechanics, Zurich, p. 131, 1926.

[17] M. D. G. Salamon MDG, "Elastic analysis of displacements and stresses induced by mining of seam or reef deposits," Part IV. J S *Afr Inst Min Metall, Part IV*, vol. 45, pp. 319-38, 1964.

[18] C. E. Massonet, "Numerical Use of Integral Procedures," *Stress Analysis-Recent Developments in Numerical and Experimental Procedures*, Wiley, New York: Zienkiewicz OC and Hollister GS, pp. 198-235, 1965.

[19] R. P. Plewman, F. H. Deist, and W. D. Ortlepp, "The development and application of a digital computer method for the solution of strata control problems," *J. S. Afr. Inst. Mining Metall*, vol. 70, pp. 33-44, 1969.

[20] S. L. Crouch, and A. M. Starfield, *Boundary Element Methods in Solid Mechanics*. London: George Allen and Unwin , 1983.

[21] P. K. Banerjee, and R. M. C. Driscoll, "Three-dimensional analysis of raked pile-groups," *Proc. Institution of Civil Engineers, Part 2*, vol. 61:653-671, 1976.

[22] F. J. Rizzo FJ, "An integral equation approach to boundary value problems of classical elastostatics," *Quarterly Journal of Applied Mathematics*, vol. 25, pp.83-95, 1967.

[23] T. A. Cruse, "Numerical solutions in three-dimensional elastostatics," *International Journal of Solids Structures*, vol. 5, pp.1259-1274, 1969.

[24] W. Thomson (Lord Kelvin), "A note on the integration of the equations of equilibrium of an elastic solid," *Cambridge and Dublin Mathematical Journal*, February 1848. Reproduced in *Physical and Mathematical Paper*, W. Thomson, Ed., Cambridge, UK: Cambridge University Press, vol. 1, pp. 97-99, 1882.

[25] E. Melan, "Der Spannungszustand der durch eine Einzelkraft im Inneren beanspruchten," *Halbscheibe Z Angew Math And Mech*, vol.12, pp. 343-346, 1932.

[26] J. C. Lachat, and J. O. Watson, "Effective numerical treatment of boundary integral equations," *Int J Num Meth Eng*, vol.10, pp. 991-1005, 1976.

[27] U. Ebrwien, C. Duenser, and W. Moser, "Efficient calculation of internal results in 2D elasticity BEM," *Engineering Analysis with Boundary Elements*, vol. 29, pp. 447-453, 2005.

[28] *Section 1, Theory and Background, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[29] *Section 2, Theory and Background, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[30] *Section 1.5, Ch. 1, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[31] *Section 1.2, Ch. 1, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[32] *Ch. 3, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[33] *Ch. 6, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[34] *Structural Elements, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[35] *Section 1.3, Ch. 1, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[36] *Section 2.7, Ch. 2, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[37] *Section 3.9, Ch. 3, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[38] *Section 2.7.3, Ch. 2, FLAC^{3D} manual version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[39] *Section 3.10.3, Ch. 3, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[40] *Section 3.3.4.2, Ch. 3, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[41] W. M. Elleithy, H. J. Al-Gahtani, and M. El-Gebeily, "Iterative coupling of BE and FE methods in elastostatics," *Eng Anal With Boundary Elem*, vol. 25, pp. 685-695, 2001.

[42] B. Helldorfer, M. Haas, and G. Kuhn, "Automatic coupling of a boundary element code with a commercial finite element system," *Advances in Engineering Software*, vol. 39, pp. 699-709, 2008.

[43] P. K. Banerjee, R. Butterfield, *Boundary Element Methods in Engineering Science*. London, UK: McGraw-Hill Book Co, 1981.

[44] M. Ameen, *Computational Elasticity: Theory of Elasticity and Finite and Boundary Element Methods*. Harrow, UK: Alpha Science International Ltd, 2005.

[45] I. A. Cermak, and P. P. Silvester, "Solution of two-dimensional field problems by boundary relaxation. Proc.," *IEEE*, vol. 155, no. 9, pp. 1341-1348, 1968.

[46] O. C. Zienkiewicz, D. W. Kelly, and P. Bettes, "The coupling of the finite element method and boundary solution procedures," *Int J Numer Methods Eng*, vol. 11, pp. 355-375, 1977.

[47] H.-B. Li, G.-M. Han, H. A. Mang, and P. Torzicky, "A new method for the coupling of finite element and boundary element discretized subdomains of elastic bodies," *Comput Methods Applied Mechanics* vol. 54, pp.161-185, 1986.

[48] S. Ganguly, "Symmetric coupling of galerkin boundary elements with finite elements," Ph.D. dissertation, Clarkson University, 1997.

[49] "State of the Art in BEM/FEM Coupling," *Int J Boundary Elem Methods Comm*., vol.4, no.2, pp.58-67, 1993.

[50] "State of the Art in BEM/FEM Coupling," *Int J Boundary Elem Methods Comm.*, vol. 4, no.3, pp.94-104, 1993.

[51] C.-C. Lin, E. C. Lawton, J. A. Caliendo, and L. R. Anderson, "An iterative finite element-boundary element algorithm," *Computers & Structures*, vol. 39, no. 5, pp. 899-909, 1996.

[52] W. L. Wendland, *On asymptotic error estimates for the combined boundary and finite element method*. Darmstadt: Fachbereich Mathematik, Technische Hochshule, 1985. Preprint Nr. 929.

[53] Y. Mitsui, Y. Ichikawa, Y. Obara, and T. Kawamato, "A coupling scheme for boundary and finite elements using a joint element," *Int J Numerical Analytical methods Geomech*, vol. 9, pp. 161-175, 1985.

[54] A. Varadarajan, K. G. Sharama, and R. B. Singh, "Some aspects of coupled FEBEM analysis of underground openings," *Int J Numer Anal Methods Geomech*, vol. 9, pp. 557-571, 1985.

[55] G. Swoboda, W. Mertz, and G. Beer, "Reological analysis of tunnel excavattions by means of coupled finite element (FEM)-boundary element (BEM) analysis," *Int J Numer Anal Methods Geomech*, vol. 11, pp.115-129, 1987.

[56] G. Beer, and J. L. Meek, "The coupling of boundary and finite element methods for infinite domain problems in elastoplasticity," in *Boundary element methods*, C. A. Brebia, Ed. Berlin: Springer, 1981, pp. 575-591.

[57] Y. T. Feng, and D. R. J. Owen, "Iterative solution of coupled FE/BE discretization for plate-foundation interaction problems," *Int J Numer Meth Engng*, vol. 39, pp.1889-1901, 1996.

[58] O. Estorff, and M. Firuziaan, "Coupled BEM/FEM approach for nonlinear soil/structure interaction," *Engineering Analysis Boundary Elements*, vol. 24, pp.715-725, 2000.

[59] S. Chen, and J. Zhao, "Modeling Tunnel Excavation Using a Hybrid DEM/BEM method," *Computer-Aided Civil and Infrastructure Engineering*, vol. 17, pp. 381-386, 2002.

[60] D. C. Rizos, and M. Wang, "Coupled BEM-FEM solutions for direct time domain soil-structure interaction analysis," *Engng Anal Boundary Elem*, vol. 26, pp. 877-888, 2002.

[61] C. A. Brebbia, and P. Georgiou, "Combination of boundary and finite elements in elastostatics," *Appl Math Model*, vol. 3, pp. 212-220, 1979.

[62] O. Tullberg, and L. Bolteus, "A critical study of different boundary element stiffness matrices," in *Boundary element methods*, C. A. Brebbia, Ed. Berlin: Springer, 1982, pp. 625-635.

[63] M. Hisatake, T. Ito, and H. Ueda, "Three dimensional symmetric coupling of boundary and finite element methods," in *Boundary elements*, C. A. Brebbia, T. Futagami, and M. Tanaka, Eds. Berlin: Springer, pp.985-994, 1983.

[64] M. A. Zarco, "Solution of soil-structure interaction problems by coupled boundary element-finite element methods," Ph.D. dissertation, Oklahoma State University, 1993.

[65] D. W. Kelly, G. Mustoe, and O. C. Zienkiewicz, "Coupling boundary element methods with other numerical methods," in *Developments in Boundary Element Methods*, vol. 1. London: Applied Science, 1979 [chapter 10].

[66] H. A. Mang, P. Torzicky, and Z. Y. Chen, "On the mechanical inconsistency of symmetrisation of unsymmetric coupling matrices for BEFEM discretizations of solids," *Comput Mechanics*, vol. 4, pp. 301-308, 1989.

[67] G. Beer, "Finite element, boundary element and coupled analysis of unbounded problems in elstostatics," *Int J Numer Methods Eng*, vol. 11, pp. 355-376, 1977.

[68] W. H. Gerstle, N. N. V. Prasad, and M. Xie, "Solution method for coupled elastostatic BEM and FEM domains," 7th International Conf. on Boundary Element Technology, Southampton: Computational Mechanics Publications, 1992, pp. 213-226.

[69] T. A. Cruse, and J. R. Osias, "Issues in merging the finite element and boundary integral equation methods," *Math Comput Modelling*, vol. 15, pp. 103-118, 1991.

[70] W. M. Elleithy, and M. Tanaka, "Interface relaxation algorithms for coupling the FEM and BEM," in *Boundary elements XXIV*, C. A. Brebbia, A. Tadu, and V. Popov, Eds. Southampton: 2002, p. 721-730.

[71] J. R. Rice, P. Tsompanopoulou, and E. A. Vavalis, "Interface relaxation methods for elliptic differential equations," *Applied Numerical Mathematics*, vol. 32, pp. 219–245, 1999.

[72] M. Mu, and J. R. Rice, "Modeling with collaborating PDE solvers—theory and practice," Comput Systems Engrg," vol. 6, pp.87–95, 1995.

[73] J. R. Rice, E. Vavalis, and D. Yang, "Analysis of a non-overlapping domain decomposition method for elliptic PDEs," *J Comput Appl Math*, vol. 87, pp. 11–19, 1998.

[74] D. Yang, "A parallel iterative non overlapping domain decomposition procedure for elliptic problems," *IMA J Numer Anal*, vol. 16, pp. 75–91, 1996.

[75] D. Yang, "A parallel domain decomposition algorithm for elliptic problems," *J Comput Math*, vol. 16, pp. 141–151, 1998.

[76] P. L. Lions, "On the Schwarz alternating method III: A variant for non-overlapping sub-domains," in *Domain Decomposition Methods for Partial Differential Equations, SIAM*, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Periaux, Eds. Philadelphia, PA, 1990, pp. 202–223.

[77] P. Le-Tallec, Y. De-Roecl, and M. Vidrascu, "Domain decomposition methods for large linearly elliptic 3 dimensional problems," *J Comput Appl Math*, vol. 34, no.1, pp. 93–117.

[78] W. M. Elleithy, and M. Tanaka, "Interface relaxation algorithms for BEM-BEM coupling and FEM-BEM coupling," *J Comput Appl Math*, vol. 3, no. 192, pp. 2977–2992, 2003.

[79] P. E. Bjorstad, and O. B. Widlund, "Iterative methods for the solution of elliptic problems in regions partitioned into substructures," *SIAM J Numer Anal*, vol. 23, no. 6, pp. 1097-1120, 1986.

[80] R. Perera, A. Ruiz, and E. Alarcon, "FEM-BEM coupling procedure through the Stelkov-Poincare's operator," in *Boundary Elements XV*, Southampton: Computational Mechanics Publications, 1993, pp. 213-226.

[81] R. Perera, and E. Alarcon, "Parallel algorithm for FE-BE coupling," in *Advances in Computational Mechanics with High Performance Computing*, B. H. V. Topping, Ed. Edinburgh: Civil-Comp Press, 1998, pp. 117-125.

[82] N. Kamiya, H. Iwase, and E. Kita, "Parallel computing for the combination method of BEM and FEM," *Engng Anal Boundary Elem*, vol. 18, pp. 221-229, 1996.

[83] N. Kamiya, and H. Iwase, "BEM and FEM combination parallel analysis using conjugate gradient and condensation," *Engng Anal Boundary Elem*, vol. 20, pp. 319-326, 1997.

[84] R. Glowinski, Q. V. Dinh, and J. Periaux, "Domain decomposition methods for nonlinear problems in fluid dynamics," *Comput Meth Appl Mech Eng*, vol. 40, pp. 27-109, 1983.

[85] J. H. Kane, D. E. Keyes, and G. K. Prasad, "Iterative solution techniques in boundary element analysis," *Int J Num Meth Eng*, vol. 31, pp. 1511-1536, 1988.

[86] I. M. Navon, and, Y. Cai, "Domain decomposition and parallel processing of a finite element model of the shallow water equations," *Comput Meth Appl Mech Eng*, vol. 106, pp. 179-212, 1993.

[87] I. S. T. Doltsinis, and S. Nolting, "Studies on parallel processing for coupled field problems," *Comput Meth Appl Mech Eng*, vol. 89, pp. 497-521, 1991.

[88] C. Y. Dong, "An iterative FE-BE coupling method for elastostatics," *Computers and Structures*, vol. 79, pp. 293-299, 2001.

[89] J. M. Ma, and M. F. Le, "A new method for coupling of boundary element method and finite element method," *Appl Math Modell*, vol. 16, pp. 43-46, 1992.

[90] D. Funaro, A. Quarteroni, and A. Zanolli, "An iterative procedure with interface relaxation for domain decomposition methods," *SIAM J Numer Anal*, vol. 25, pp. 1213-1236, 1988.

[91] E. Stein E, and M. Kreienmeyer, "Coupling of BEM and FEM by a multiplicative Schwarz method and its parallel implementation," *Engrg Comput*, vol. 15, no. 2, pp. 173–198, 1998.

[92] W. M. Elleithy, H. J. Al-Gahtani, "An overlapping domain decomposition approach for coupling the finite and boundary element methods," *Engng Anal Boundary Elements*, vol. 24, no. 5, pp. 391–398, 2000.

[93] P. K. Banerjee, R. M. C. Driscoll, "Three-dimensional analysis of raked pile groups," in Proc Inst of Civ Engrs, *Res and Theory*, 1976, vol. 61, pp. 653-671.

[94] P. K. Banerjee, and T. G. Davies, "Analysis of some reported case histories of laterally loaded pile groups," in *Int Conf Num Meth in Offshore Piling*, Institute of Civil Engineers, London, 1979, pp. 101-108.

[95] T. G. Davies, "Linear and nonlinear analysis of pile groups," Ph.D. thesis, Univ. of Wales, Univ. College, Cardiff, 1979.

[96] J. T. Katsikadelis, and A. E. Armenakas, "A new boundary equation solution to the plate problem," *J of Appl Mech*, Transactions of the ASME, vol. 56, pp. 364-374, 1989.

[97] R. Rangogni, and M. Reali, "The coupling of the finite difference method and the boundary element method," *Appl Math Model*, vol. 6, pp. 233-236, 1982.

[98] H. Wei, G. Wang, |Z. Wang, "Coupling Computation of the BEM and FDM in 3D Capacitance Extraction," presented at 4th International Conference on ASIC Proc, Chinese Inst Electr, IEEE Beijing Sect, China, 2001, pp.716-719.

[99] C. A. Brebbia, and S. Walker, *Boundary Element Techniques in Engineering*. London: Newnes-Butterworth, 1980.

[100] D. A. Curran, M. Cross, and B. A. Lewis, "Solution of parabolic differential equations by the boundary element method using discretization in time," *Applied Mathematical Modelling*, vol. 4, pp. 398–400, 1980.

[101] C. P. Hong, T. Umeda, and Y. Kimura, Numerical models for casting solidification: Part I. The coupling of the boundary element and finite difference methods for solidification problems. Metallurgical Transactions B 1984;15B:91-99.

[102] M. Jamali, "BEM modeling of surface water wave motion with laminar boundary layers," *Engineering Analysis with Boundary Elements*, vol. 30, pp. 14–21, 2006.

[103] S. J. DeSilva, and C. L. Chan, "Coupled boundary element method and finite difference method for the heat conduction in laser processing," *Applied Mathematical Modelling*, vol. 32, pp. 2429–2458, 2008.

[104] J. C. Wu, and J. F. Thomson, "Numerical solutions of time-dependent incompressible Navier-Stokes equations using an integro-differential formulation," *Computers and fluids*, vol. 1, pp. 197-215, 1973.

[105] J. C. Wu, "Numerical boundary conditions for viscous flow problems," *AIAA J*, vol. 14, no.8, pp.1042-1049, 1976.

[106] J. C. Wu, A. Sugavanam, "Method of numerical solution of turbulent flow problems," AIAA J, vol. 16, pp. 948-955, 1978.

[107] N. Sankar, and J. C. Wu, "Viscous flow around oscillating airfoil-a numerical study," AIAA 11th Fluid and Plasma Dynamics Conf, Seattle, Washington, 1978.

[108] G. Coulmy, and T. S. Luu, "Solution of the Navier-Stokes equation for an incompressible flow by an integro-differential method," in Proc Inst Symp on Innovative Num Analysis in *Appl Engng Sci*, Versailles, France, 1977.

[109] J. F. Tompson, "Two approaches to the three-dimensional jet-in-cross wind problem: a vortex lattice model and a numerical solution of the Navier-Stokes equations," Ph.D. thesis, presented at Georgia Institute of Technology, Atlanta, Georgia, 1971.

[110] P. Ghadimi, and A. Dashtimanesh, "Solution of 2D Navier–Stokes equation by coupled finite difference-dual reciprocity boundary element method," *Applied Mathematical Modelling*, vol. 35, pp. 2110–2121, 2011.

[111] S. M. Mamoon, and P. K. Banjeree, "Time-domain analysis of dynamically loaded single piles," *J of Engng Mech*, vol. 118, no. 1, pp.140-160, 1992.

[112] D. Mazzoni, and U. Kristiansen, "Finite Difference Method for the Acoustic Radiation of an Elastic Plate Excited by a Turbulent Boundary layer: A Spectral Domain Solution," *Flow, Turbulence and Combustion*, vol. 61, pp. 133–159, 1999.

[113] F. Janod, and O. Coutant, "Seismic response of three-dimensional topographies using a time-domain boundary element method," *Geophys J Int*, vol. 142, pp. 603-614, 2000.

[114] R. Naserizadeh, H. B. Bingham, and A. Noorzad, "A coupled boundary element-finite difference solution of the elliptic modified mild slope equation," *Engineering Analysis with Boundary Elements*, vol. 35, pp. 25–33, 2011.

[115] S. Biswas, and R. W. Snidle, "Calculation of surface deformation in point contact elasto-hydrodynamics," *J Lubrication Technol*, vol. 99, pp. 313-317, 1977.

[116] D. J. S. Barrett, A. El-Zafrany, I. R. W. McLuckie, "Elasto-hydrodynamic analysis of bearings by coupling boundary element and finite difference methods," in International Series on Advances in Boundary Elements, *Boundary Elements XXIV*, vol.13, pp. 711-719, 2002.

[117] W. H. Peng, G. H. Cao, S. C. Li and Dongzhengzhu, "Darcy-Stokes equations with finite difference and natural boundary element coupling method," *Computer modeling in engineering and sciences*, vol. 75, no.3, pp.173-188, 2011.

[118] M. He, P. J. Bishop, A. J. Kassab, and A. Minardi, "A Coupled FDM/BEM solution for the conjugate heat transfer problem," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 28, no.2, pp. 139-154, 1995.

[119] M. S. Ingber, C. C. Schmidt, J. A. Tanski, and J. Phillips, "Boundary-element analysis of 3D diffusion problems using a parallel domain decomposition method," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 44, no. 2, pp. 145-164, 2003.

[120] A. J. Davies, and D. Crann, "Laplace transform time domain-decomposition for diffusion problems," *Computer Modeling in Engineering and Sciences*, vol. 18, no. 2, pp. 1-4, 2007.

[121] K. L. Leung, P. B. Zavareh, and D. E. Beskos, "2D elastostatic analysis by a symmetric BEM/FEM scheme," *Engineering Analysis with Boundary Elements*, vol. 15, pp. 67-78, 1995.

[122] K. Kohno, T. Tsunada, H. Seto H, and M. Tanaka, "Hybrid stress analysis of boundary and finite elements by a super-element method," *Finite Elements in Analysis and Design*, vol. 7, pp. 279-290, 1991.

[123] U. Gohner, and H. Mauch, "FLOTRAN: Numerical method and industrial applications," *International Journal of Computer Applications in Technology*, vol. 11, pp.199-202, 1998.

[124] El-Gebeily M, W. Elleithy, H. J. Al-Gahtani, "Convergence of the domain decomposition finite element–boundary element coupling methods," *Comput Methods Appl Mech Engrg*, vol. 191, pp. 4851–4867, 2002.

[125] S. Timoshenko, and J. N. Goodier, *Theory of Elasticity*. Toronto: MacGraw-Hill, 3rd ed., pp. 396-398, 1970.

[126] D. L. Holl, "Stress transmission in earths," Proc High Res Board, 1940, vol. 20, pp. 709-721.

[127] H. G. Poulos, and E. H. Davis, *Elastic solutions for soil and rock mechanics*. New York, NY: J. Wiley & Sons Inc., 1974, pp. 54-57.

[128] R. D. Mindlin, "Stress distribution around a hole near the edge of a plate under tension," Proc Soc Exptl Stress Anal, vol. 5, pp. 56-68, 1948.

[129] *Section 3.2.3, Ch. 3, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[130] *Section 1, Verification Problems, FLAC^{3D} manual, version 3*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2005.

[131] *Section 4, FISH IN FLAC^{3D}, FLAC^{3D} manual, version 4*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2009.

[132] F. Tonon, E. Pan, and B. Amadei, "Green's functions and BEM formulations for 3D anisotropic media," Computer and Structures, vol. 79, no.5, pp. 469–482, 2000.

[133] X. –W. Gao, T. G. Davies, *Boundary Element Programming in Mechanics*. Cambridge, UK: Cambridge Univ. Press, 2002.

[134] *Verification Problems, FLAC3D manual, version 4*, Itasca Consulting Group Inc., Minneapolis, Minnesota, 2009.

[135] W. –F. Chen, Bearing Capacity of Square, *Rectangular and Circular Footings, in Limit Analysis and Soil Plasticity, Developments in Geotechnical Engineering 7*. New York, NY: Elsevier Scientific Publishing Co., 1975. Ch. 7, pp. 295-340.

[136] W.S. Venturini, *Boundary element method in geomechanics*. Berlin, Springer-Verlag1983.

Finite Difference-Boundary Element Methods in Infinite and Semi-infinite Media in Geomechanics

Volume 2

by

Ziad Halabi

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Civil Engineering

Waterloo, Ontario, Canada, 2013

Table of Contents

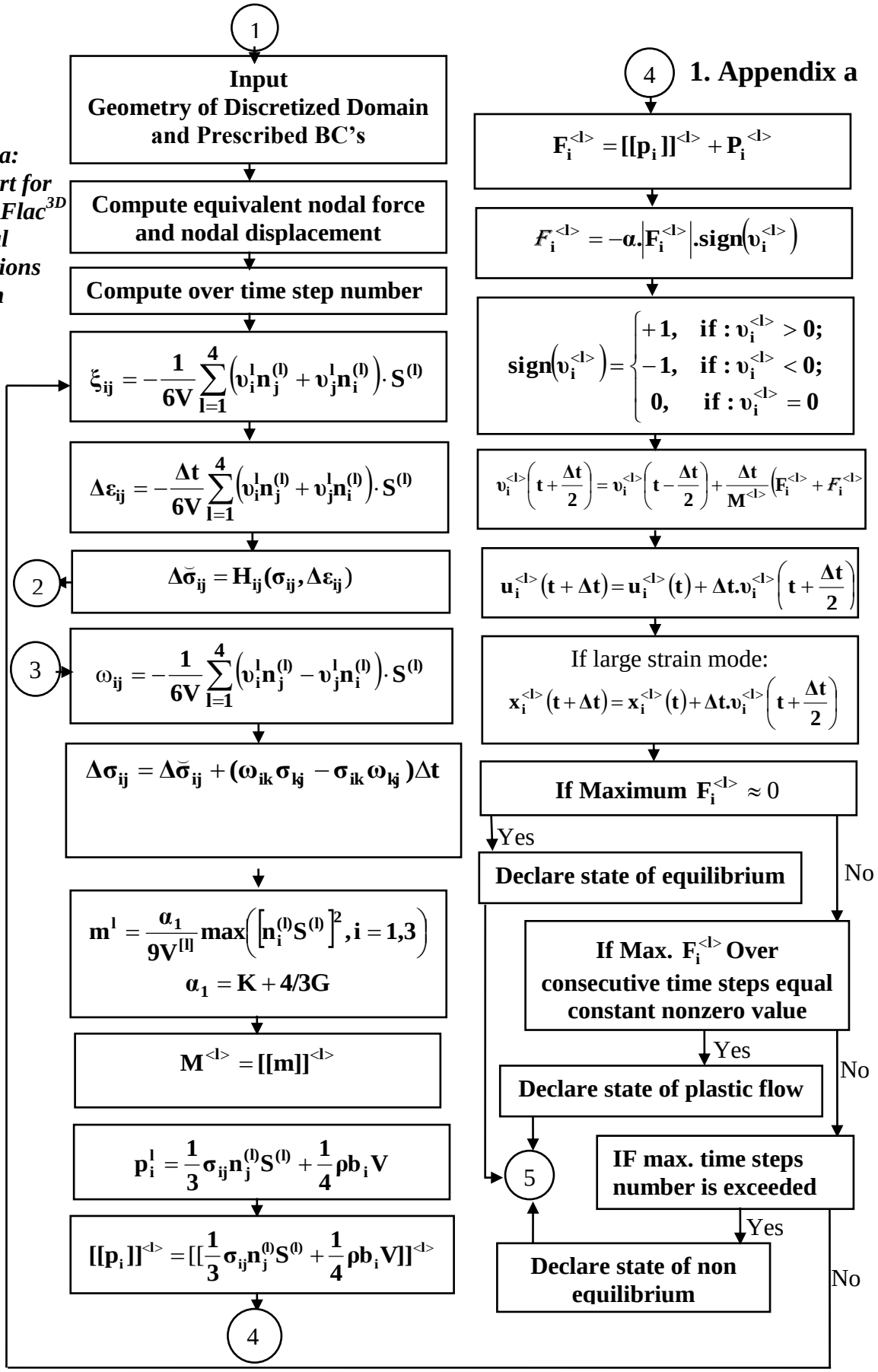


Figure 3.a: Flow chart for the main Flac^{3D} numerical computations algorithm

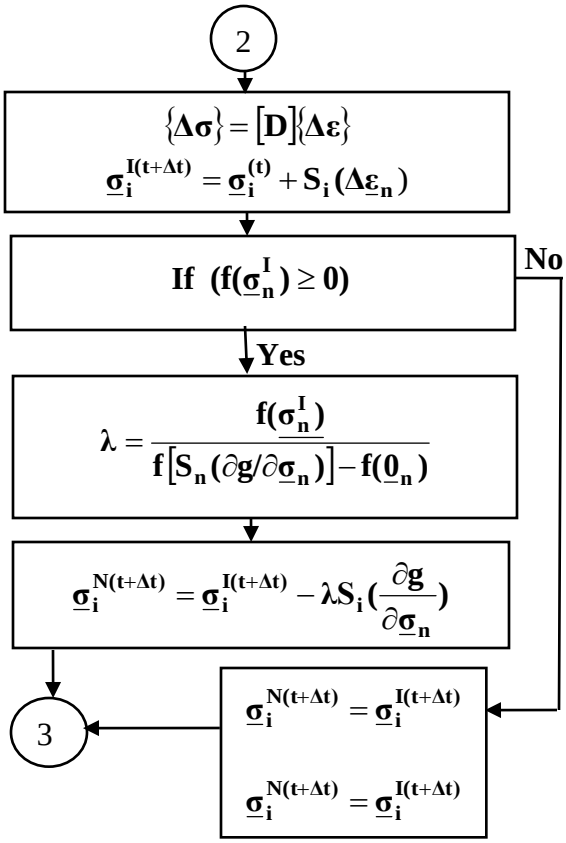


Figure 3.b Flow chart for the stress increment computations in an elastic-plastic constitutive model

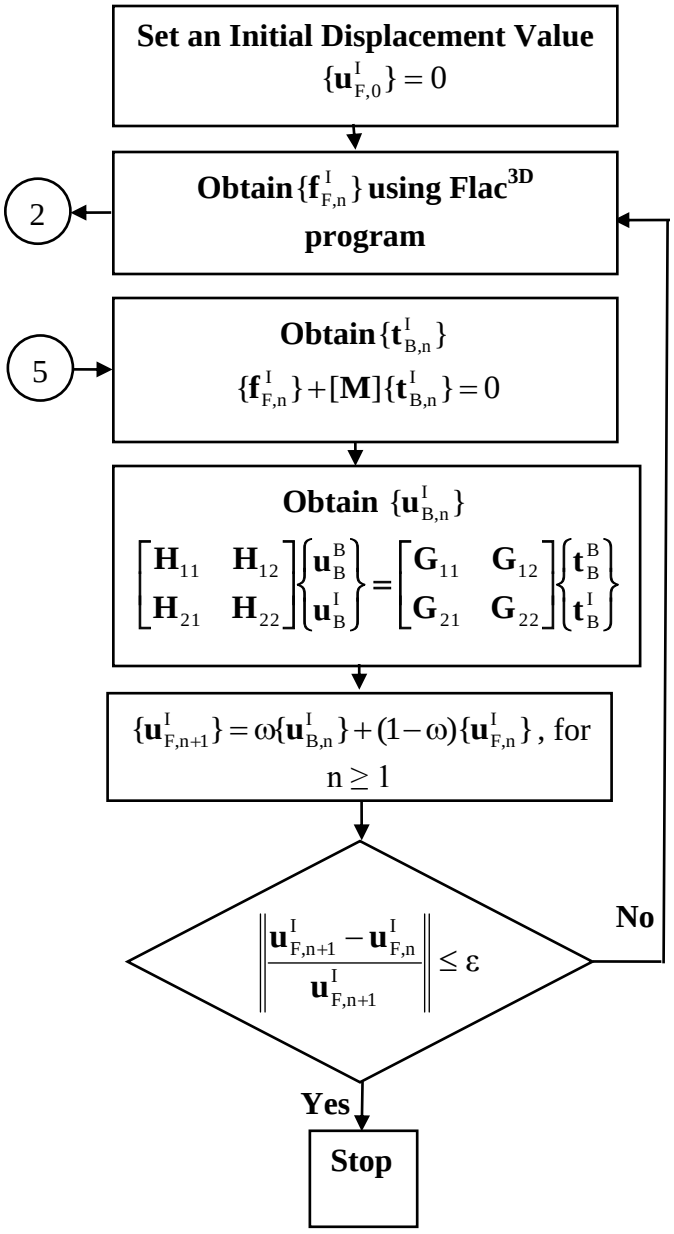


Figure 5.a Flow chart for the first coupling method (IDDM) algorithm

2. Appendix b

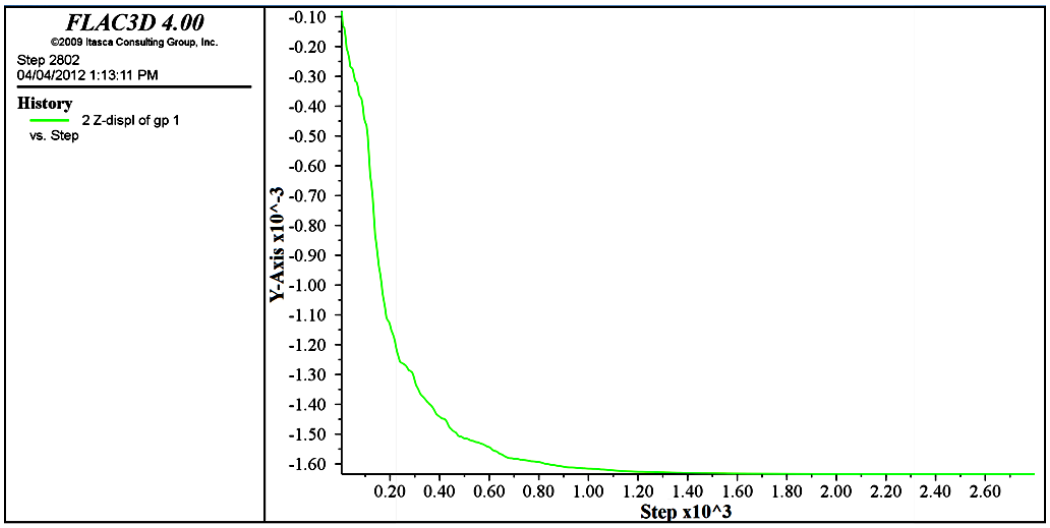


Figure 5.4. Vertical displacement u_Z at the top of the spherical excavation obtained by Flac3D with a model of outer radius R = 2 m, *(Y-Axis ≡ u_Z)*.

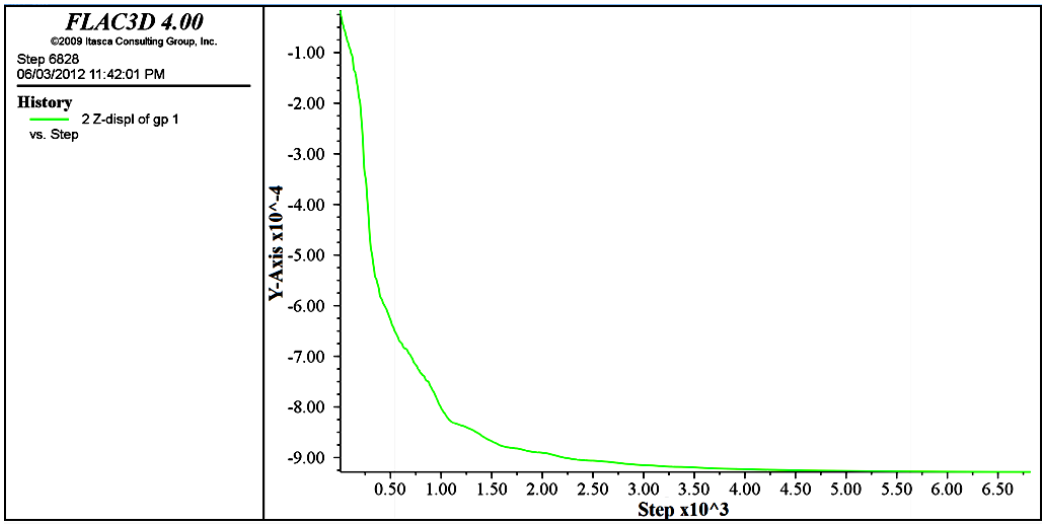


Figure 5.5. Vertical displacement u_Z at the top of the spherical excavation obtained by Flac3D with a model of outer radius R = 12 m, *(Y-Axis ≡ u_Z)*.

Table 5.3 Exact and Flac3D normalized stresses $\sigma z/\sigma_{field}$ in plane z = 0 in terms of r/a

R/a	r/a	Falc3D	Exact Solution	Relative Error%
2	1.06	2.2180	1.7194	29.00
	1.19	1.7878	1.4468	23.58
	1.31	1.5372	1.2939	18.80
	1.43	1.3682	1.2026	13.77
	1.56	1.2399	1.1451	8.28
	1.68	1.1322	1.1073	2.25
	1.81	1.0338	1.0815	-4.41
	1.93	0.9373	1.0633	-11.85
3	1.04	1.8759	1.7833	5.19
	1.14	1.6239	1.5322	5.98
	1.24	1.4504	1.3635	6.37
	1.36	1.3297	1.2498	6.40
	1.49	1.2445	1.1728	6.11
	1.64	1.1826	1.1203	5.56
	1.79	1.1359	1.0844	4.76
	1.97	1.0988	1.0596	3.70
	2.16	1.0672	1.0424	2.38
	2.36	1.0383	1.0303	0.77
	2.59	1.0098	1.0218	-1.18
	2.85	0.9801	1.0158	-3.51

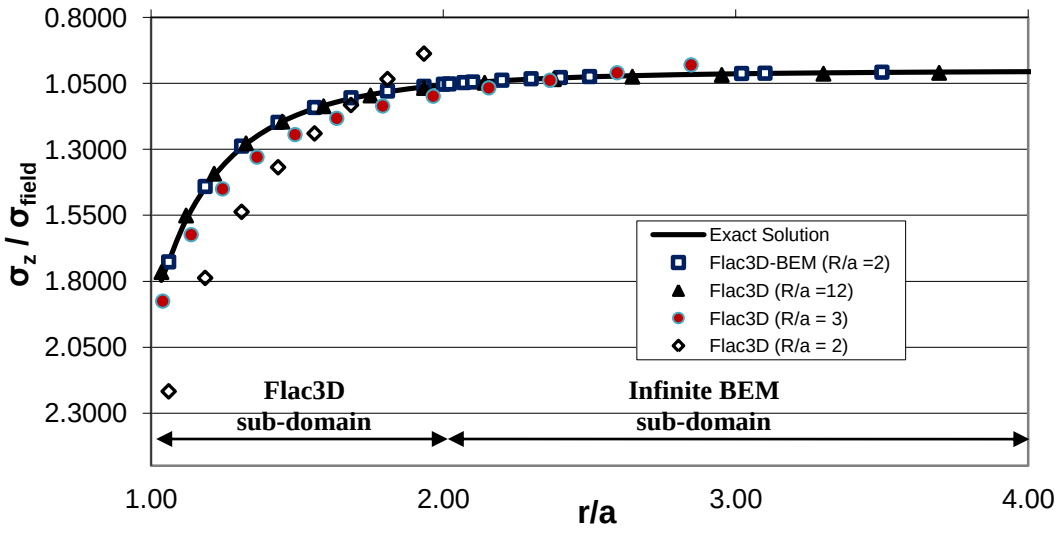


Figure 5.17 Coupled Flac^{3D} -BEM, uncoupled Flac^{3D} and exact vertical stress (σ_z) in plane (z = 0) as a function of (r/a) in both Flac^{3D} and BEM sub-domains

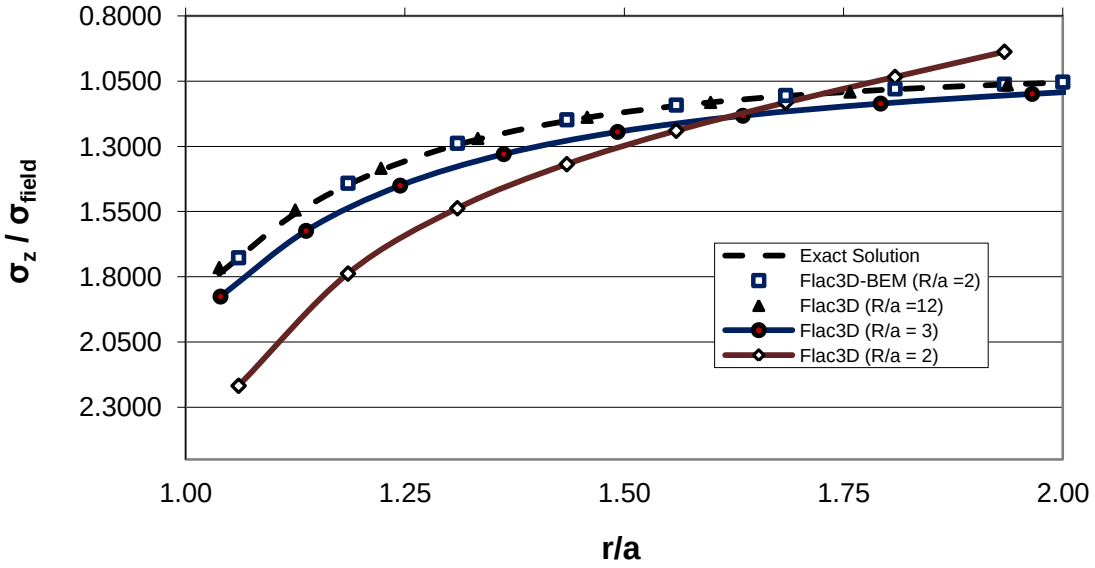


Figure 5.18 Coupled Flac^{3D} -BEM, uncoupled Flac^{3D} and exact vertical stress (σ_z) in plane (z = 0) as a function of (r/a) in Flac^{3D} sub-domain

Table 5.5 Exact and Flac^{3D} normalized stresses σ_z/σ_{field} in plane z = 0 in terms of r/a

R/a	r/a	Exact solution	Falc^{3D}	R. E %
8	1.0356	1.7969	1.7657	-1.7368
	1.1199	1.5683	1.5512	-1.0926
	1.2158	1.4009	1.3927	-0.5853
	1.3249	1.2803	1.2776	-0.2074
	1.4490	1.1945	1.1952	0.0582
	1.5902	1.1343	1.1369	0.2338
	1.7508	1.0923	1.0961	0.3423
	1.9334	1.0633	1.0676	0.4036
	2.1412	1.0434	1.0479	0.4327
	2.3775	1.0297	1.0343	0.4405
	2.6464	1.0204	1.0248	0.4336
	2.9522	1.0140	1.0182	0.4160
	3.3000	1.0096	1.0135	0.3892
	3.6957	1.0066	1.0101	0.3532
	4.1458	1.0045	1.0076	0.3068
	4.6577	1.0031	1.0056	0.2478
	5.2401	1.0021	1.0039	0.1727
	5.9025	1.0015	1.0023	0.0770
	6.6560	1.0010	1.0006	-0.0450
	7.5132	1.0007	0.9987	-0.2009
12	1.0382	1.7882	1.7646	-1.3195
	1.1249	1.5577	1.5439	-0.8860
	1.2227	1.3916	1.3839	-0.5513
	1.3331	1.2733	1.2694	-0.3055
	1.4578	1.1899	1.1883	-0.1339
	1.5985	1.1315	1.1313	-0.0197
	1.7574	1.0910	1.0916	0.0530
	1.9368	1.0629	1.0639	0.0969
	2.1394	1.0435	1.0448	0.1219
	2.3680	1.0301	1.0315	0.1350
	2.6262	1.0209	1.0224	0.1406
	2.9177	1.0145	1.0160	0.1415
	3.2468	1.0101	1.0115	0.1394
	3.6183	1.0071	1.0084	0.1354
	4.0378	1.0049	1.0062	0.1298
	4.5114	1.0035	1.0047	0.1225
	5.0461	1.0024	1.0036	0.1134
	5.6497	1.0017	1.0027	0.1020
	6.3312	1.0012	1.0021	0.0878
	7.1006	1.0008	1.0015	0.0699
	7.9693	1.0006	1.0011	0.0475
	8.9501	1.0004	1.0006	0.0191
	10.0573	1.0003	1.0001	-0.0168
	11.3074	1.0002	0.9996	-0.0622

Table 5.6 Exact and coupled Flac^{3D} -BEM normalized stress σ_z/σ_{field} in plane (z = 0) in terms of r/a

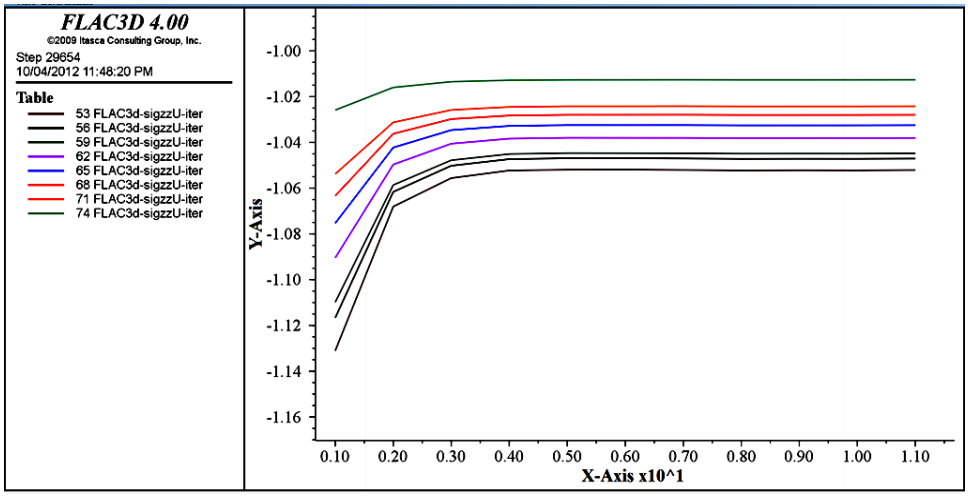
R /a = 2	r/a	Exact solution	Flac^{3D} -BEM	R. E. %
Flac^{3D} sub-domain	1.0603	1.7194	1.7270	0.4410
	1.1850	1.4468	1.4411	-0.3915
	1.3098	1.2939	1.2880	-0.4604
	1.4345	1.2026	1.1979	-0.3941
	1.5593	1.1451	1.1415	-0.3122
	1.6840	1.1073	1.1046	-0.2439
	1.8088	1.0815	1.0792	-0.2076
	1.9335	1.0633	1.0613	-0.1859
BEM sub-domain	2.00	1.0558	1.0534	-0.2292
	2.02	1.0538	1.0521	-0.1599
	2.07	1.0491	1.0471	-0.1967
	2.10	1.0466	1.0448	-0.1704
	2.20	1.0393	1.0381	-0.1200
	2.30	1.0335	1.0325	-0.0945
	2.40	1.0287	1.0280	-0.0764
	2.50	1.0249	1.0242	-0.0631
	3.02	1.0129	1.0126	-0.0297
	3.10	1.0118	1.0116	-0.0271
	3.50	1.0079	1.0077	-0.0179
	4.02	1.0050	1.0049	-0.0115
	4.10	1.0047	1.0046	-0.0108
	4.50	1.0035	1.0034	-0.0080
	5.00	1.0025	1.0024	-0.0058
	6.00	1.0014	1.0014	-0.0033
	8.00	1.0006	1.0006	-0.0014
	10.00	1.0003	1.0003	-0.0007
	12.00	1.0002	1.0002	-0.0004
	13.00	1.0001	1.0001	-0.0003

Table 5.7 Coupled Flac^{3D} -BEM (right) and uncoupled Flac^{3D} (left) horizontal displacement u_x in plane (z = 0) as a function of (r/a)

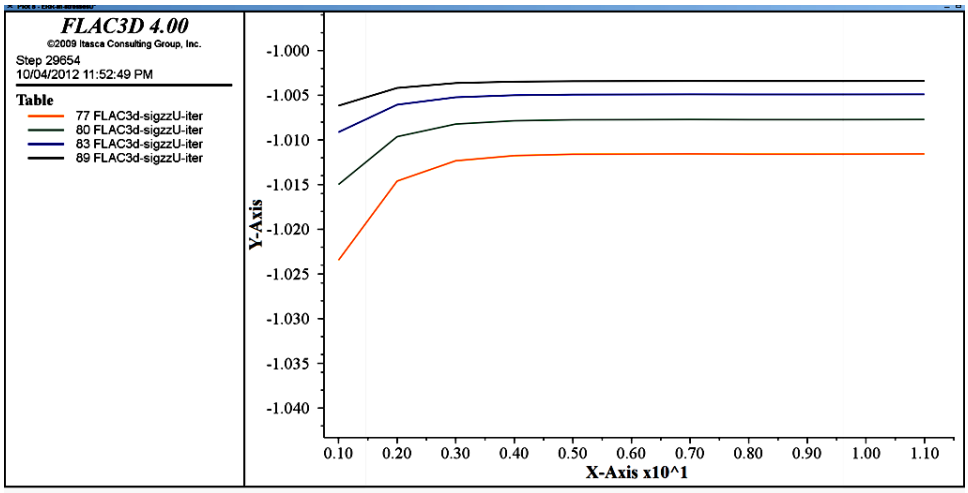
R/a	r/a	u_x/mm
8	1.0000	0.211257
	1.0792	0.207451
	1.1693	0.196897
	1.2718	0.181807
	1.3883	0.164087
	1.5209	0.145291
	1.6717	0.126605
	1.8433	0.108869
	2.0385	0.092621
	2.2604	0.078160
	2.5130	0.065596
	2.8002	0.054912
	3.1269	0.046003
	3.4985	0.038719
	3.9213	0.032887
	4.4022	0.028330
	4.9492	0.024883
	5.5714	0.022399
	6.2791	0.020752
	7.0842	0.019843
	8.0000	0.019597
12	1.0000	0.209872
	1.0816	0.205752
	1.1737	0.194678
	1.2777	0.179146
	1.3951	0.161185
	1.5276	0.142351
	1.6773	0.123778
	1.8462	0.106230
	2.0370	0.090177
	2.2523	0.075854
	2.4955	0.063339
	2.7700	0.052585
	3.0799	0.043483
	3.4298	0.035875
	3.8248	0.029593
	4.2708	0.024464
	4.7743	0.020325
	5.3428	0.017027
	5.9846	0.014441
	6.7092	0.012453
	7.5273	0.010968
	8.4509	0.009909
	9.4936	0.009210
	10.6709	0.008824
	12.0000	0.008714

R /a = 2	r/a	u_x/mm
Flac^{3D} sub-domain	1.0000	0.207099
	1.1250	0.198067
	1.2500	0.179957
	1.3750	0.160600
	1.5000	0.142568
	1.6250	0.126544
	1.7500	0.112637
	1.8750	0.100652
	2.0000	0.090337
BEM sub-domain	2.0200	0.088018
	2.0700	0.084267
	2.1000	0.082186
	2.2000	0.075761
	2.3000	0.070001
	2.4000	0.064836
	2.5000	0.060198
	3.0200	0.042389
	3.1000	0.040350
	3.5000	0.032035
	4.0200	0.024538
	4.1000	0.023619
	4.5000	0.019711
	5.0000	0.016044
	6.0000	0.011213
	8.0000	0.006347
	10.0000	0.004074
	12.0000	0.002833
	13.0000	0.002416

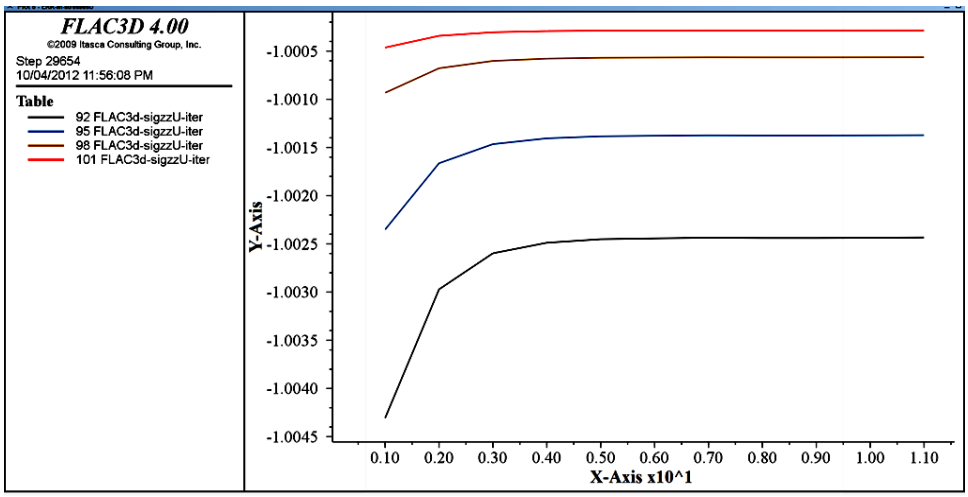
a.


FLAC3D 4.00
©2009 Itasca Consulting Group, Inc.
Step 29654
10/04/2012 11:48:20 PM
Table
53 FLAC3d-sigzzU-iter
56 FLAC3d-sigzzU-iter
59 FLAC3d-sigzzU-iter
62 FLAC3d-sigzzU-iter
65 FLAC3d-sigzzU-iter
68 FLAC3d-sigzzU-iter
71 FLAC3d-sigzzU-iter
74 FLAC3d-sigzzU-iter
Y-Axis
-1.00
-1.02
-1.04
-1.06
-1.08
-1.10
-1.12
-1.14
-1.16
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
1.10
X-Axis x10^1

b.


FLAC3D 4.00
©2009 Itasca Consulting Group, Inc.
Step 29654
10/04/2012 11:52:49 PM
Table
77 FLAC3d-sigzzU-iter
80 FLAC3d-sigzzU-iter
83 FLAC3d-sigzzU-iter
89 FLAC3d-sigzzU-iter
Y-Axis
-1.000
-1.005
-1.010
-1.015
-1.020
-1.025
-1.030
-1.035
-1.040
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
1.10
X-Axis x10^1

c.


FLAC3D 4.00
©2009 Itasca Consulting Group, Inc.
Step 29654
10/04/2012 11:56:08 PM
Table
92 FLAC3d-sigzzU-iter
95 FLAC3d-sigzzU-iter
98 FLAC3d-sigzzU-iter
101 FLAC3d-sigzzU-iter
Y-Axis
-1.0005
-1.0010
-1.0015
-1.0020
-1.0025
-1.0030
-1.0035
-1.0040
-1.0045
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
1.10
X-Axis x10^1

d.

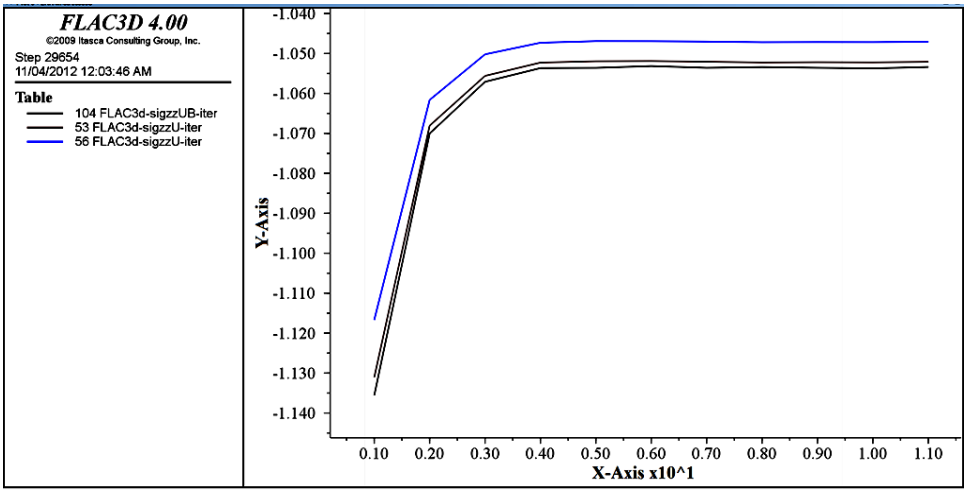


Figure 5.31a. b. c &d. Convergence of stress (σ_z) at chosen points in the BEM sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ σ_Z).*

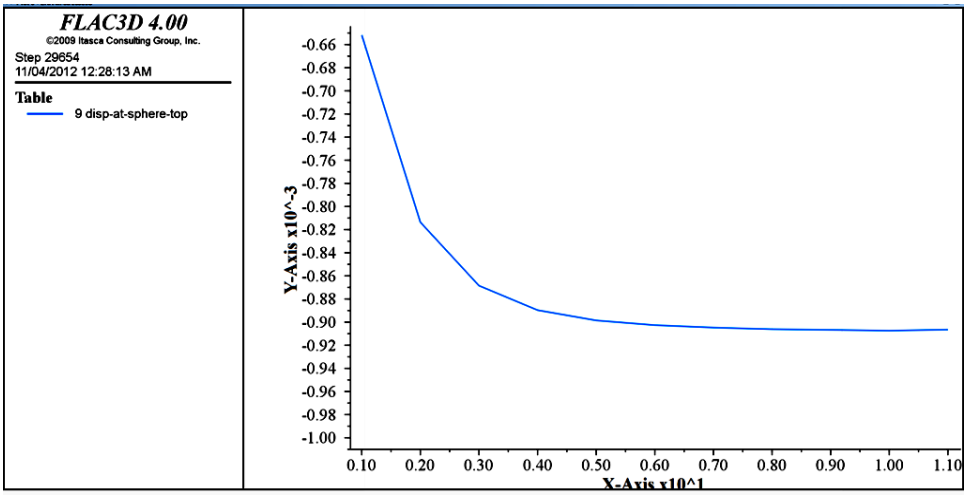
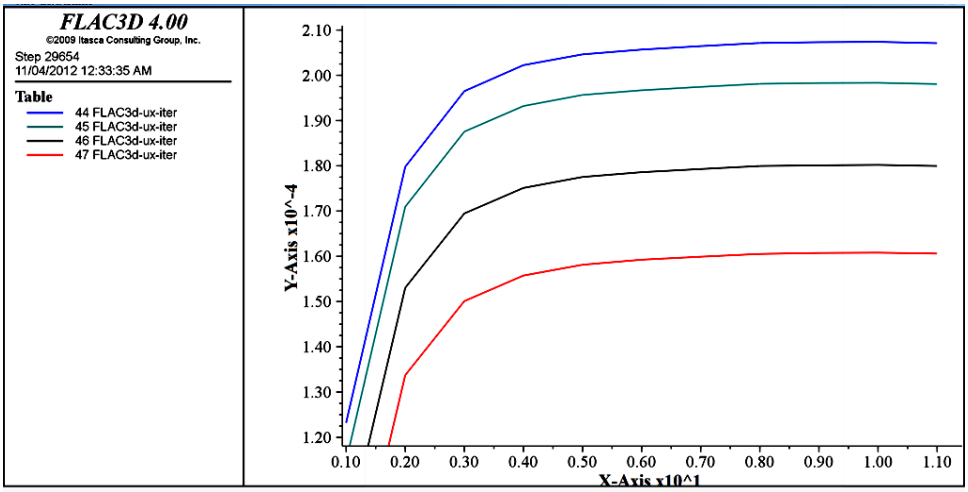


Figure 5.32 Convergence of vertical displacement (u_z) at point (A ≡ 9) in the Flac3D sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ u_Z).*

a.



b.

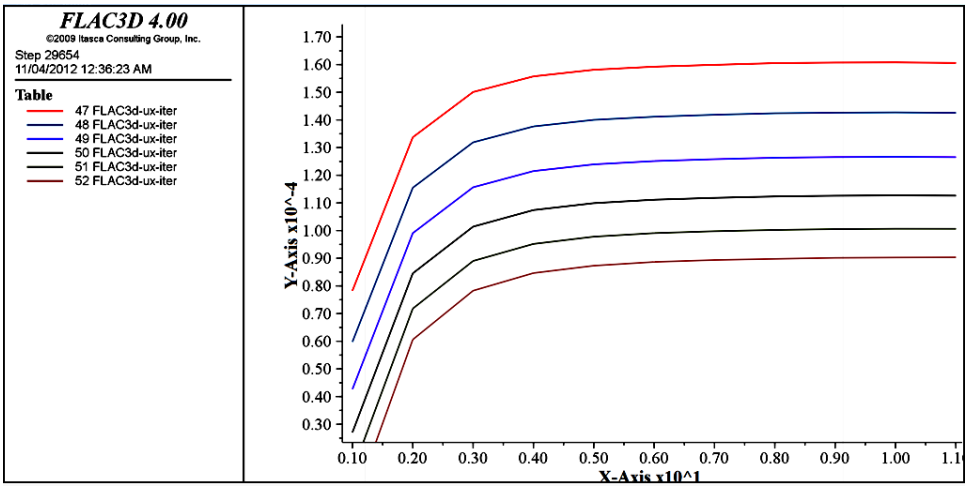
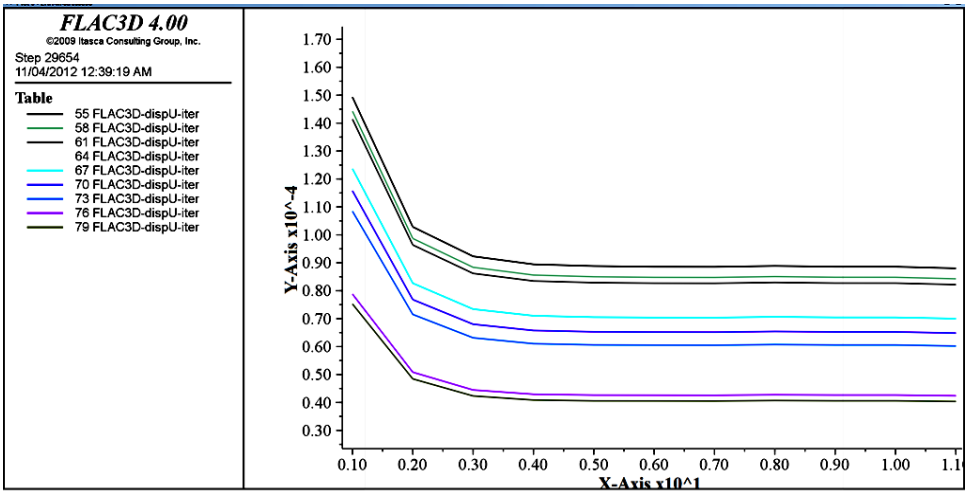
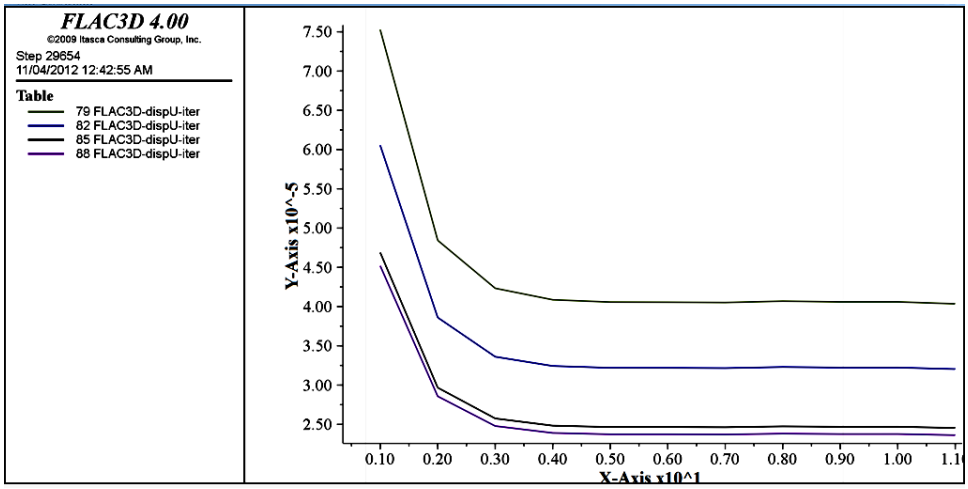


Figure 5.33 Convergence of horizontal displacement (u_x) at chosen points in the Flac3D sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ u_x).*

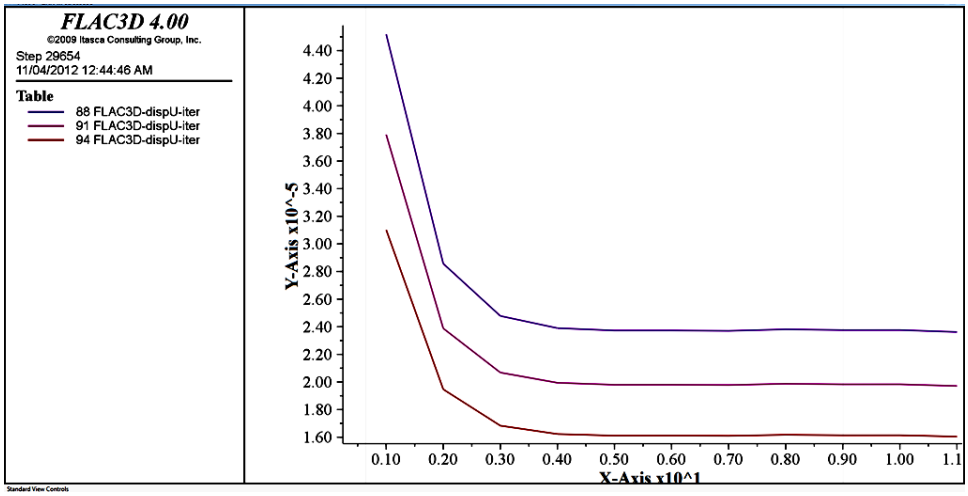
a.



b.



c.



d.

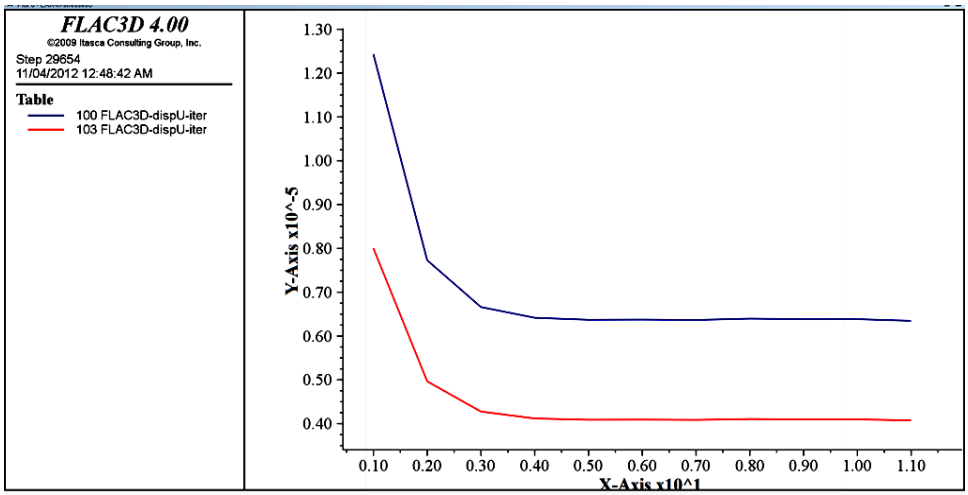


Figure 5.34 Convergence of horizontal displacement (u_x) at chosen points in the BEM sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ u_x).*

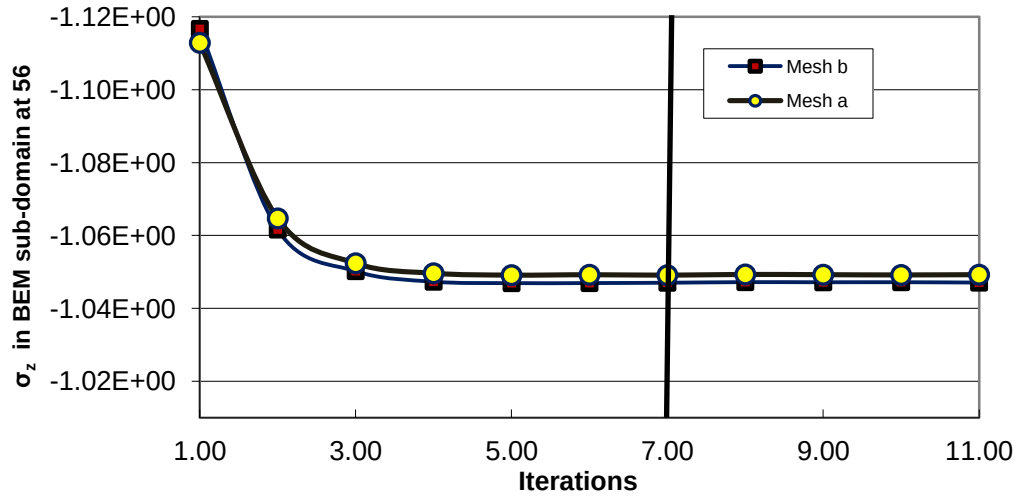


Figure5. 36 Evolution of stress (σ_z) at zone No. 56 in the BEM sub-domain over the iterative scheme for meshes a and b.

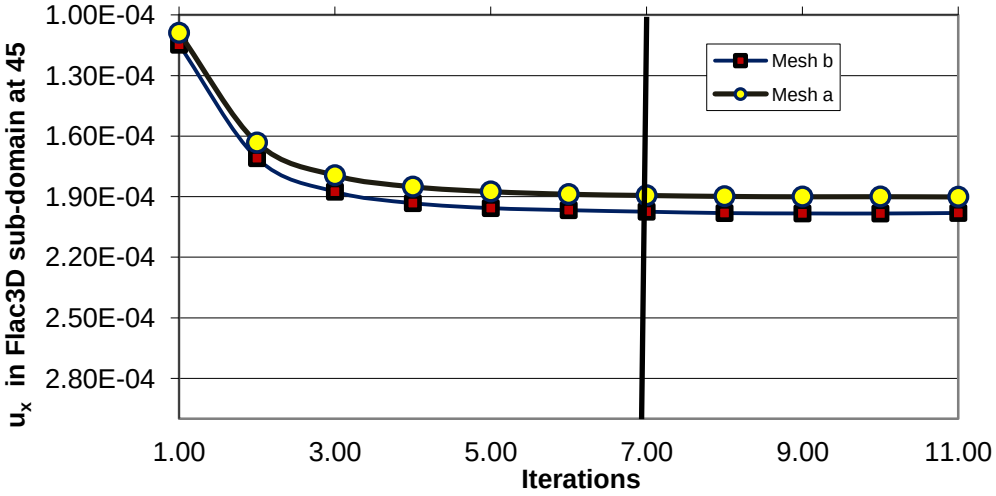


Figure5. 37 Evolution of horizontal displacement (u_x) at point 45 in the Flac^{3D} sub-domain over the iterative scheme for meshes a and b.

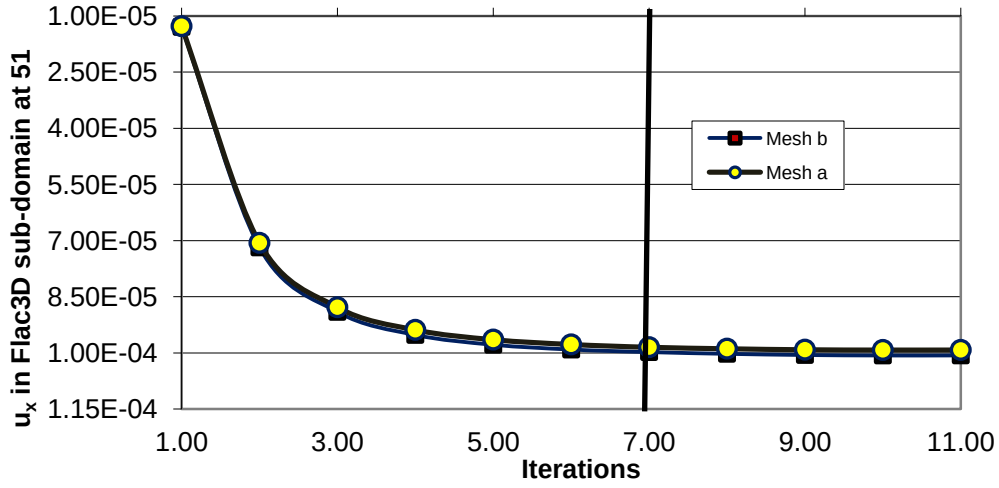


Figure5. 38 Evolution of horizontal displacement (u_x) at point 51 in the Flac^{3D} sub-domain over the iterative scheme for meshes a and b.

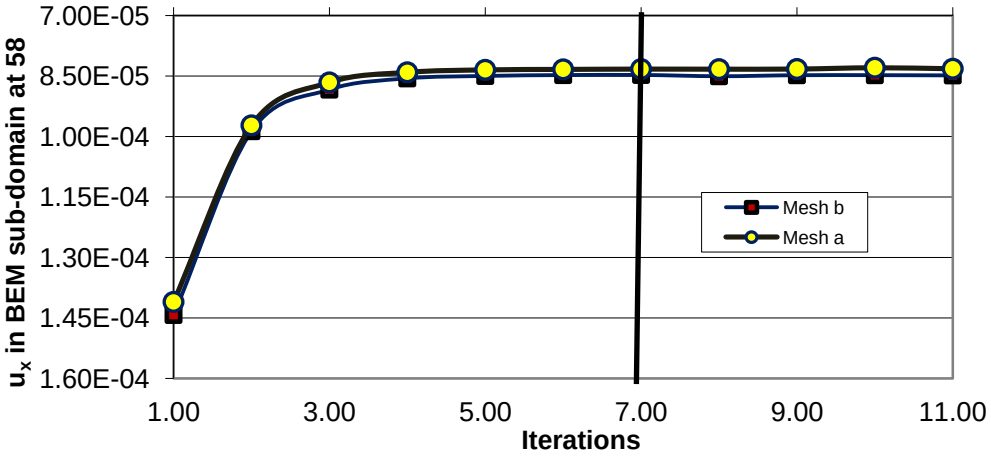


Figure5.39 Evolution of horizontal displacement (u_x) at point 58 in the BEM sub-domain over the iterative scheme for meshes a and b.

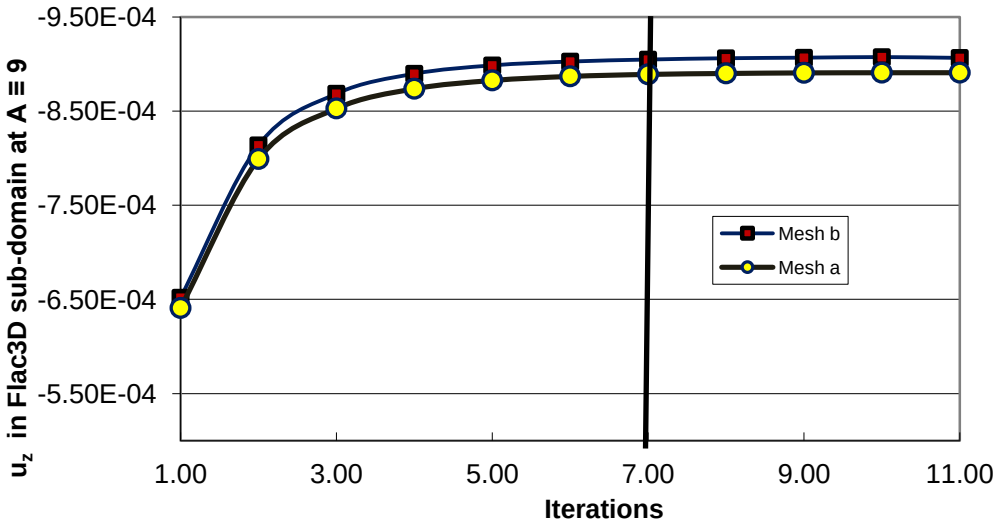


Figure5.40 Evolution of vertical displacement (u_z) at point (A ≡ 9) in the Flac^{3D} sub-domain over the iterative scheme for meshes a and b.

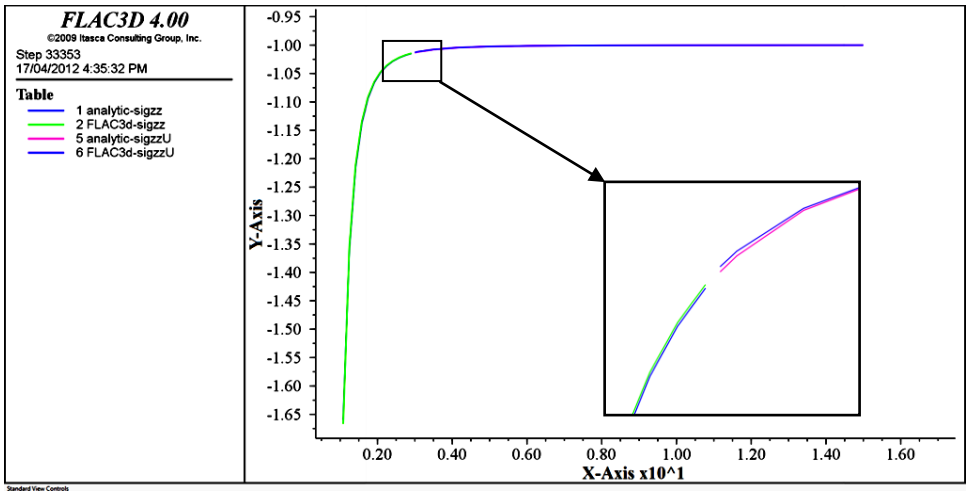
Table 5.11 Exact and coupled normalized stresses σ_z/σ_{field} in plane z = 0

R /a = 3	r/a	Exact Solution	Flac^{3D} -BEM	R. E. %
Flac^{3D} sub-domain	1.0817	1.6599	1.6671	0.4368
	1.2481	1.3592	1.3555	-0.2712
	1.4145	1.2145	1.2110	-0.2833
	1.5809	1.1374	1.1350	-0.2145
	1.7473	1.0930	1.0914	-0.1515
	1.9137	1.0658	1.0647	-0.1047
	2.0802	1.0482	1.0475	-0.0751
	2.2466	1.0364	1.0359	-0.0561
	2.4130	1.0282	1.0278	-0.0420
	2.5794	1.0223	1.0219	-0.0348
	2.7458	1.0179	1.0176	-0.0315
	2.9122	1.0146	1.0143	-0.0302
BEM sub-domain	3.00	1.0132	1.0128	-0.0443
	3.10	1.0118	1.0114	-0.0403
	3.50	1.0079	1.0077	-0.0176
	4.02	1.0050	1.0049	-0.0097
	4.10	1.0047	1.0046	-0.0090
	4.50	1.0035	1.0034	-0.0065
	5.00	1.0025	1.0024	-0.0046
	6.00	1.0014	1.0014	-0.0027
	8.00	1.0006	1.0006	-0.0011
	10.00	1.0003	1.0003	-0.0006
	12.00	1.0002	1.0002	-0.0003
	13.00	1.0001	1.0001	-0.0003

Table 5.12 Exact and coupled normalized stresses σ_z/σ_{field} in plane z = 0

R /a = 4	r/a	Exact Solution	Flac^{3D} -BEM	R. E. %
Flac^{3D} sub-domain	1.0933	1.6303	1.6446	0.8811
	1.2807	1.3226	1.3224	-0.0159
	1.4681	1.1846	1.1829	-0.1398
	1.6555	1.1147	1.1131	-0.1364
	1.8429	1.0759	1.0747	-0.1089
	2.0304	1.0528	1.0519	-0.0818
	2.2178	1.0382	1.0375	-0.0622
	2.4052	1.0285	1.0280	-0.0485
	2.5926	1.0219	1.0215	-0.0388
	2.7800	1.0172	1.0168	-0.0316
	2.9674	1.0137	1.0135	-0.0254
	3.1549	1.0112	1.0109	-0.0210
	3.3423	1.0092	1.0090	-0.0191
	3.5297	1.0077	1.0075	-0.0176
	3.7171	1.0065	1.0063	-0.0157
	3.9045	1.0055	1.0054	-0.0141
BEM sub-domain	4.00	1.0051	1.0049	-0.0160
	4.10	1.0047	1.0046	-0.0139
	4.50	1.0035	1.0034	-0.0074
	5.00	1.0025	1.0024	-0.0044
	6.00	1.0014	1.0014	-0.0022
	8.00	1.0006	1.0006	-0.0009
	10.00	1.0003	1.0003	-0.0005
	12.00	1.0002	1.0002	-0.0003
	13.00	1.0001	1.0001	-0.0002

a.



b.

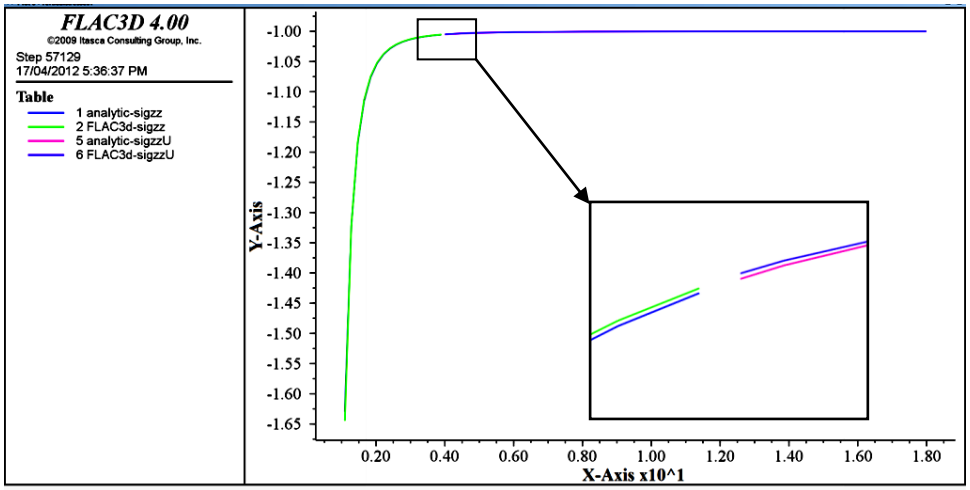
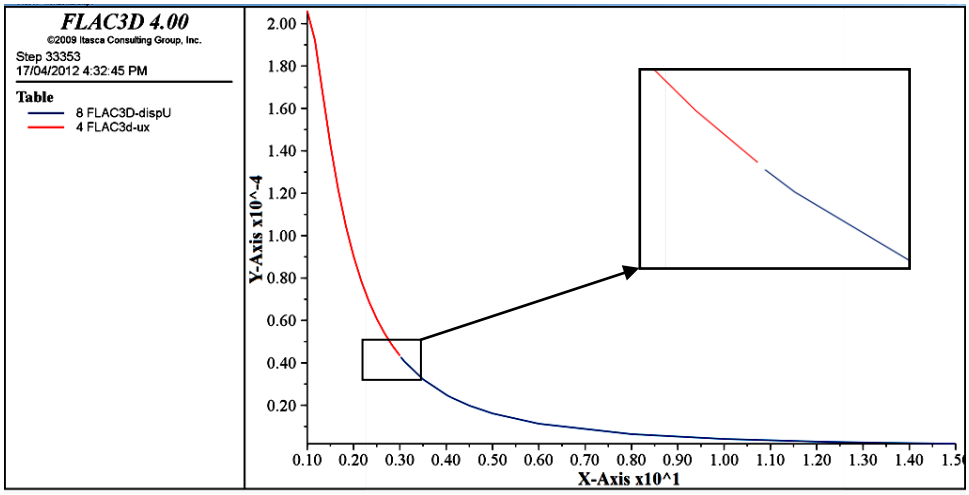
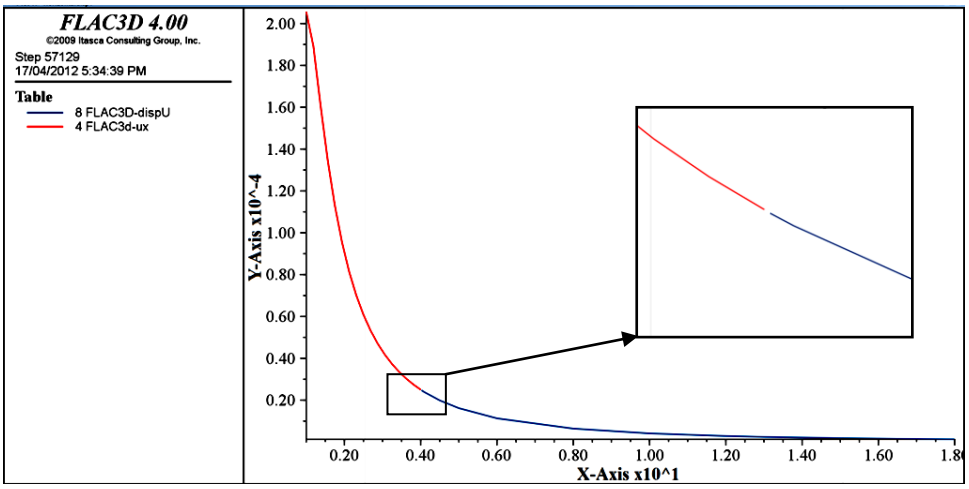


Figure 5.42 Coupled and exact vertical stress (σ_z) in plane (z = 0 in both Flac3D and BEM sub-domains, *(X-Axis ≡ r) and (Y-Axis ≡ σ_Z)*. The Flac3D models ratio: a. R/a = 3, b. R/a = 4



a.



b.

Figure 5.46 The compatibility of coupled horizontal displacement u_x in plane (z = 0) across the interface, (X-Axis ≡ r) and (Y-Axis ≡ u_x). Flac3D model ratios: a. R/a = 3, b. R/a = 4

Table 5.16 Vertical stress relative error in plane (z = 0) after applying the second method of coupled Flac^{3D} -BEM, E = 1000 & 2000 MPa, ν = 0.0

R /a = 2	r/a	R. E. %
Flac^{3D} sub-domain	1.0523	2.7546
	1.1761	4.4098
	1.2999	5.4641
	1.4237	6.1343
	1.5475	6.5680
	1.6713	6.8462
	1.7951	7.0228
	1.9190	7.1320
BEM sub-domain	2.0000	6.9814
	2.0700	6.2083
	2.1000	5.7712
	2.2000	4.7362
	2.3000	3.9504
	2.4000	3.3141
	2.5000	2.7968
	3.0200	1.2805
	3.1000	1.1512
	3.5000	0.7072
	4.0200	0.4118
	4.1000	0.3818
	4.5000	0.2685
	5.0000	0.1820
	6.0000	0.0948
	6.0200	0.0937
	6.1000	0.0895
	6.5000	0.0718
	7.0000	0.0557
	8.0000	0.0356
	8.0700	0.0346
	8.1000	0.0342
	8.5000	0.0291
	9.0000	0.0242
	10.0000	0.0172

R /a = 3	r/a	R. E. %
Flac^{3D} sub-domain	1.0760	0.4100
	1.2416	1.5279
	1.4071	2.1475
	1.5727	2.4846
	1.7382	2.6650
	1.9038	2.7582
	2.0693	2.7964
	2.2348	2.8014
	2.4004	2.7851
	2.5659	2.7543
	2.7315	2.7130
	2.8970	2.6623
BEM sub-domain	3.0000	2.5535
	3.1000	2.2386
	3.5000	1.3290
	4.0200	0.7406
	4.1000	0.6821
	4.5000	0.4645
	5.0000	0.3034
	6.0000	0.1489
	6.0200	0.1470
	6.1000	0.1398
	6.5000	0.1100
	7.0000	0.0836
	8.0000	0.0515
	8.0700	0.0500
	8.1000	0.0493
	8.5000	0.0416
	9.0000	0.0340
	10.0000	0.0237
	11.0000	0.0172
	12.0000	0.0129
	13.0000	0.0099

Table 5.16 Vertical stress relative error in plane (z = 0) after applying the second method of coupled Flac^{3D} -BEM, E = 1000 & 2000 MPa, ν = 0.0

R /a = 4	r/a	R. E. %
Flac^{3D} sub-domain	1.0896	0.2903
	1.2763	0.7226
	1.4631	0.9924
	1.6499	1.1517
	1.8367	1.2413
	2.0235	1.2899
	2.2102	1.3131
	2.3970	1.3208
	2.5838	1.3198
	2.5838	1.3201
	2.7706	1.3114
	2.9574	1.2969
	2.9574	1.2973
	3.1441	1.2797
	3.3309	1.2592
	3.5177	1.2369
	3.5177	1.2380
	3.7045	1.2137
	3.7045	1.2132
	3.8913	1.1868
BEM sub-domain	4.0000	1.1497
	4.1000	1.0362
	4.5000	0.6920
	5.0000	0.4409
	6.0000	0.2062
	8.0000	0.0663
	8.1000	0.0632
	8.5000	0.0527
	9.0000	0.0426
	10.0000	0.0290
	11.0000	0.0206
	12.0000	0.0152
	13.0000	0.0116

Table 5.18.b Vertical stress relative error in plane (z = 0) by applying the Second Method/SDDM of coupled Flac^{3D} -BEM, E = 1000 MPa, v = 0.3

R /a = 2	r/a	R. E. %
Flac^{3D} sub-domain	1.0273	7.7201
	1.0896	8.3301
	1.1518	8.7813
	1.2141	9.0969
	1.2763	9.3015
	1.3386	9.4134
	1.4009	9.4502
	1.4631	9.4264
	1.5254	9.3540
	1.5876	9.2384
	1.6499	9.0881
	1.7122	8.9103
	1.7744	8.7081
	1.8367	8.4821
	1.8989	8.2340
	1.9612	7.9591
BEM sub-domain	2.0000	6.2327
	2.0700	5.4708
	2.1000	5.1435
	2.2000	4.3120
	2.3000	3.6395
	2.4000	3.0907
	2.5000	2.6433
	3.1000	1.1789
	3.5000	0.7586
	4.1000	0.4340
	4.5000	0.3150
	5.0000	0.2206
	6.0000	0.1208
	7.0000	0.0735
	8.0000	0.0481
	9.0000	0.0332
	10.0000	0.0239

R /a = 3	r/a	R. E. %
Flac^{3D} sub-domain	1.0474	3.8386
	1.1472	4.1924
	1.2469	4.4235
	1.3467	4.5557
	1.4464	4.6136
	1.5462	4.6168
	1.6459	4.5781
	1.7457	4.5089
	1.8454	4.4165
	1.9452	4.3081
	2.0449	4.1841
	2.1447	4.0475
	2.2444	3.8997
	2.3442	3.7420
	2.4439	3.5767
	2.5437	3.4023
	2.6434	3.2190
	2.7432	3.0267
	2.8430	2.8251
	2.9427	2.6105
BEM sub-domain	3.0000	2.2052
	3.1000	1.9587
	3.5000	1.2158
	4.1000	0.6668
	4.5000	0.4729
	5.0000	0.3234
	6.0000	0.1709
	6.5000	0.1301
	7.0000	0.1014
	8.0000	0.0652
	8.5000	0.0535
	9.0000	0.0445
	10.0000	0.0318

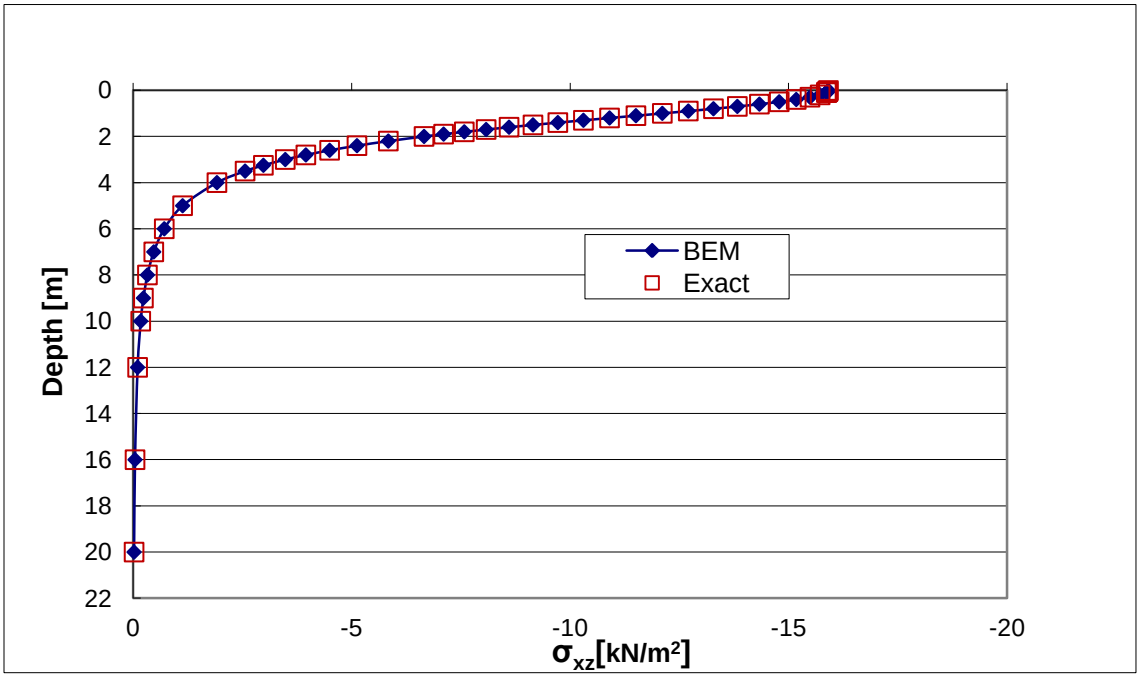
Table 5.18.b Vertical stress relative error in plane (z = 0) by applying the Second Method/SDDM of coupled Flac^{3D} -BEM, E = 1000 MPa, ν = 0.3

R /a = 4	r/a	R. E. %
Flac^{3D} sub-domain	1.0607	2.0249
	1.1855	2.1280
	1.3102	2.2153
	1.4350	2.2742
	1.5598	2.3057
	1.6846	2.3112
	1.8094	2.2989
	1.9342	2.2727
	2.0589	2.2344
	2.1837	2.1889
	2.3085	2.1379
	2.4333	2.0804
	2.5581	2.0185
	2.6829	1.9517
	2.8077	1.8799
	2.9324	1.8050
	3.0572	1.7274
	3.1820	1.6467
	3.3068	1.5617
	3.4316	1.4736
	3.5564	1.3815
	3.6811	1.2858
	3.8059	1.1854
	3.9307	1.0789
BEM sub-domain	4.0000	0.9596
	4.5000	0.6119
	5.0000	0.4086
	6.0000	0.2079
	7.0000	0.1200
	8.0000	0.0756
	8.5000	0.0616
	9.0000	0.0508
	10.0000	0.0358

R /a = 6	r/a	R. E. %
Flac^{3D} sub-domain	1.0879	1.0855
	1.2662	0.7477
	1.4446	0.6809
	1.6229	0.6923
	1.8013	0.7136
	1.9796	0.7291
	2.1580	0.7369
	2.3363	0.7380
	2.5147	0.7338
	2.6930	0.7250
	2.8714	0.7142
	3.0497	0.7014
	3.2280	0.6857
	3.4064	0.6682
	3.5847	0.6501
	3.7631	0.6303
	3.9414	0.6094
	4.1198	0.5876
	4.2981	0.5644
	4.4765	0.5404
	4.6548	0.5151
	4.8331	0.4886
	5.0115	0.4613
	5.1898	0.4328
	5.3682	0.4028
	5.5465	0.3719
	5.7249	0.3399
	5.9032	0.3060
BEM sub-domain	6.0000	0.2850
	7.0000	0.1588
	8.0000	0.0960
	9.0000	0.0624
	10.0000	0.0429

3. Appendix c

a.



b.

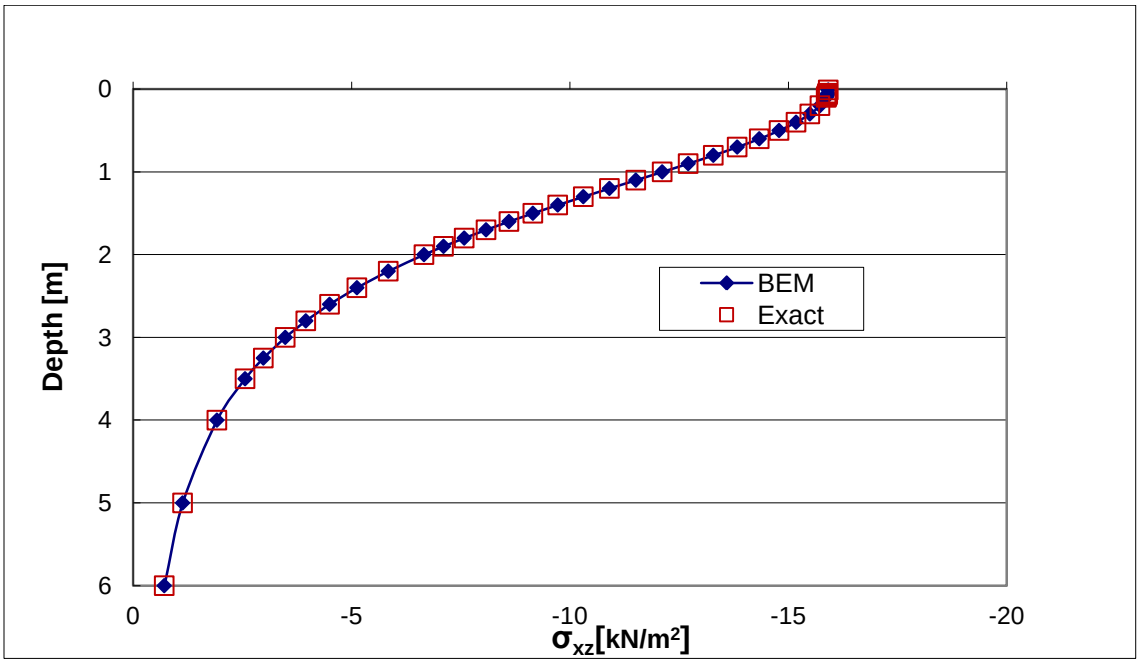
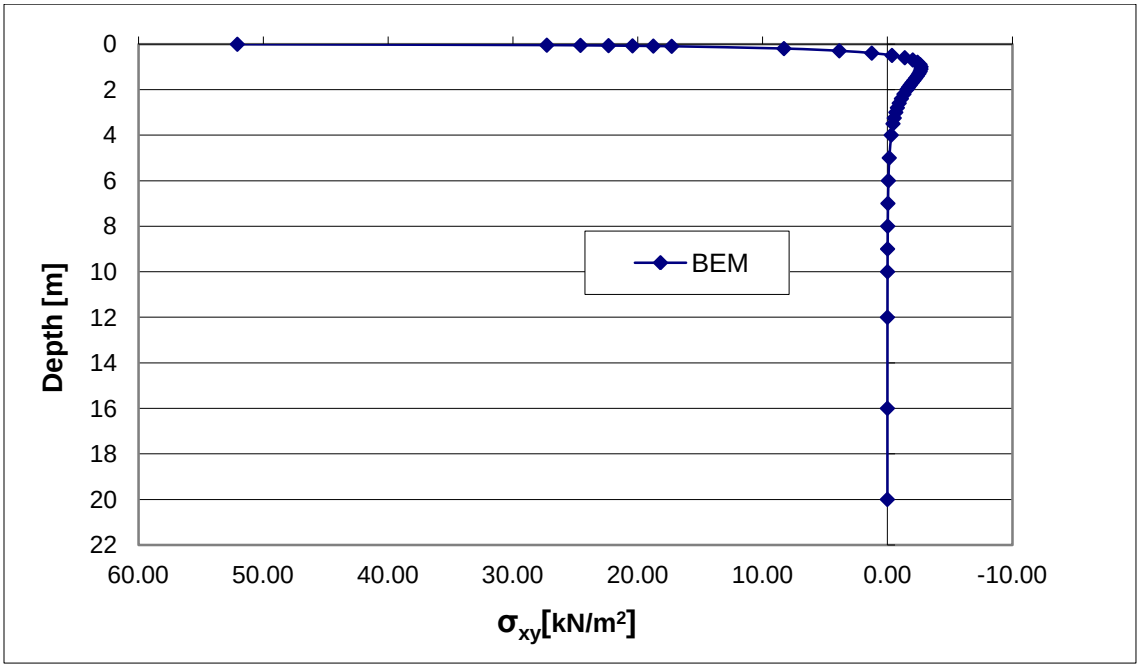


Figure 5.63 a. & b. The shear stress σ_{xz} under the corner of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0$].

a.



b.

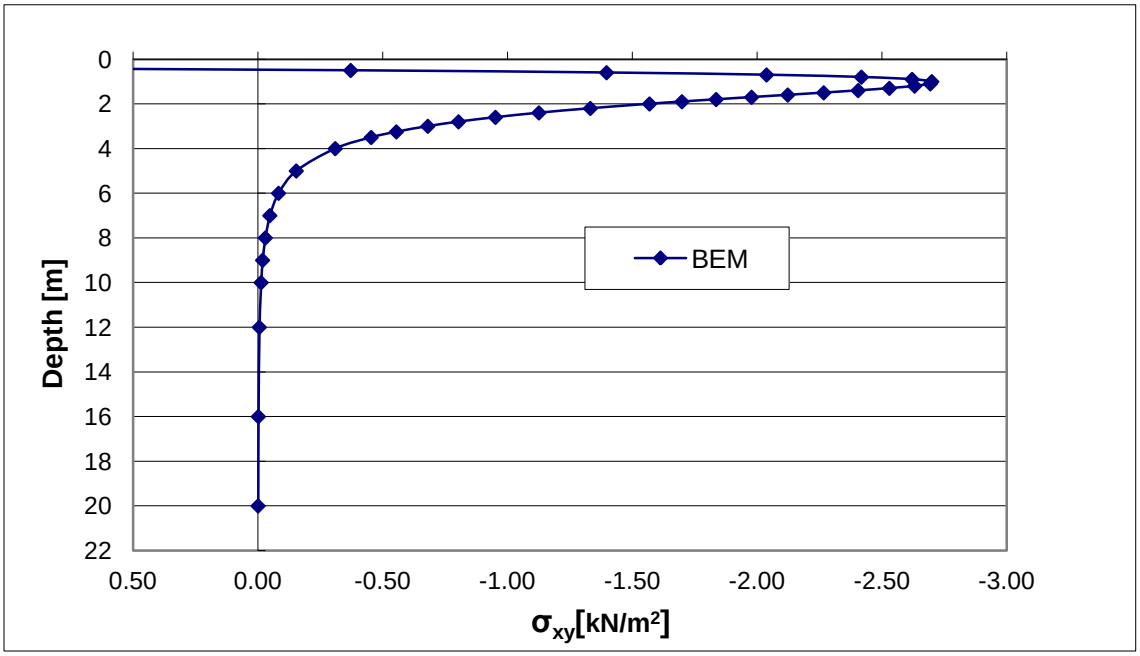
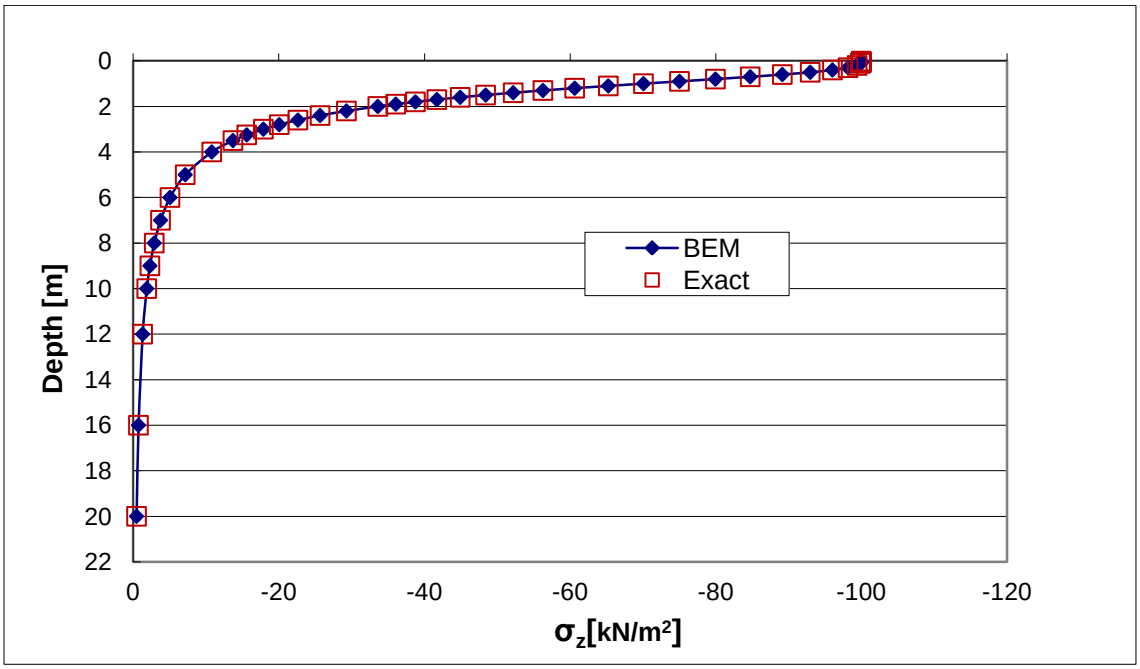


Figure 5.64 a. & b. The shear stress σ_{xy} under the corner of a uniform square load [Mindlin's BEM solution only, $\upsilon = 0$].

a.



b.

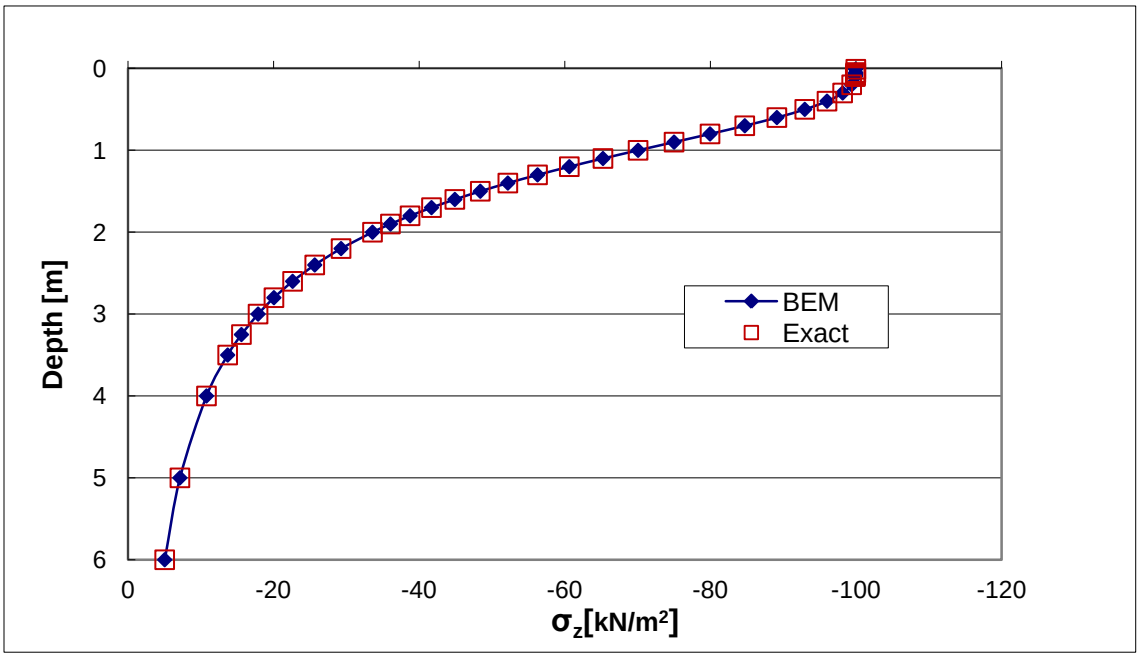
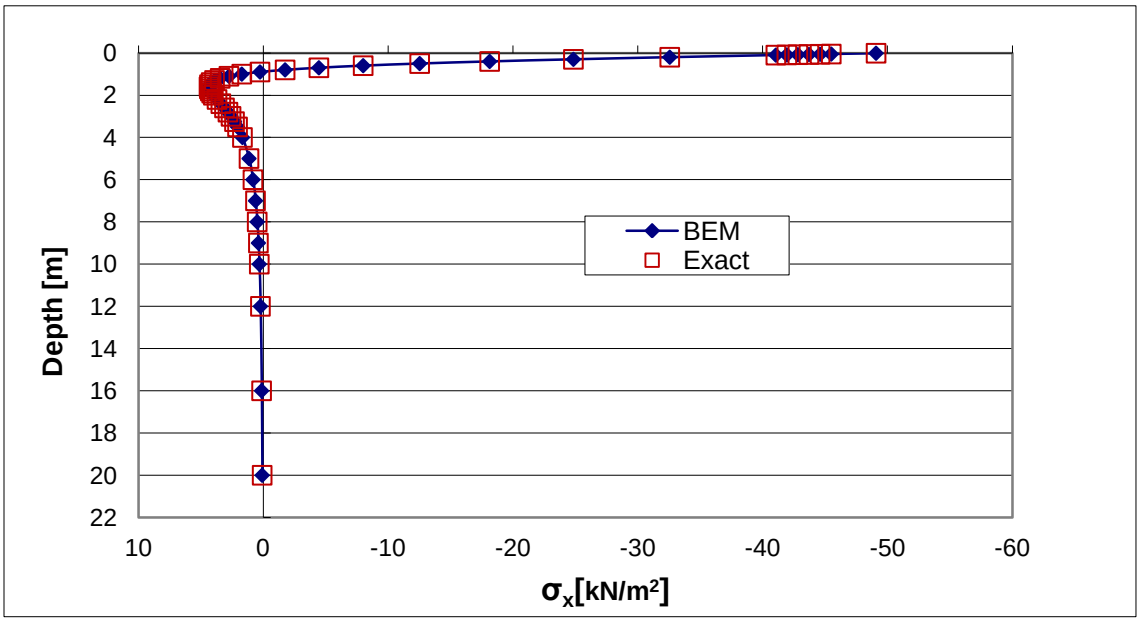


Figure 5.65 a. & b. The vertical stress σ_z under the center of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0$].

a.



b.

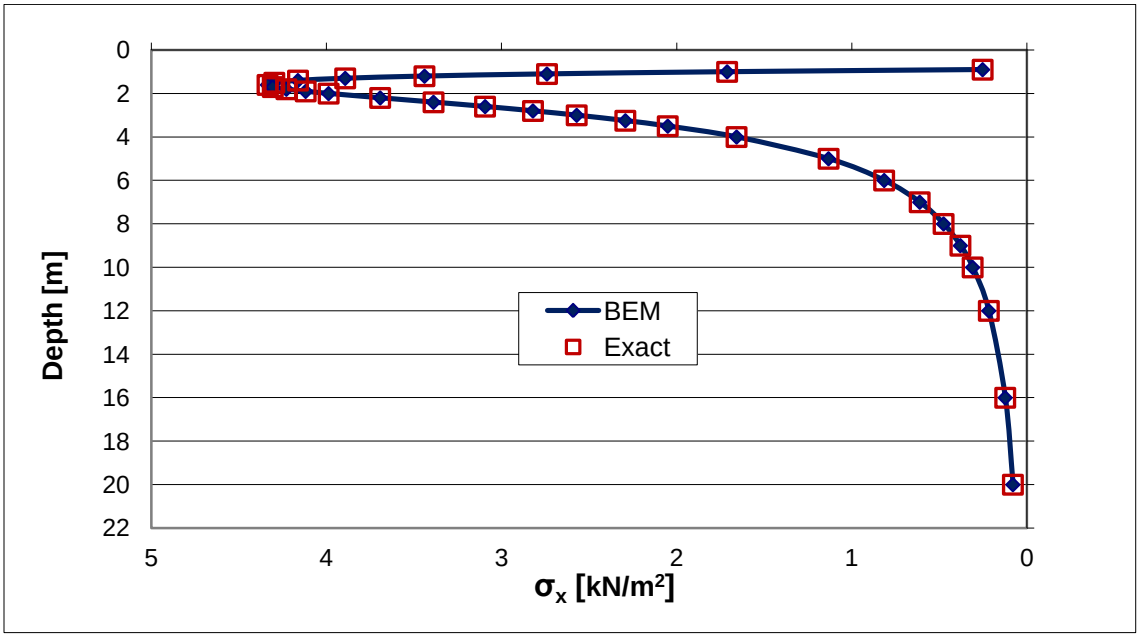


Figure 5.66 a. & b. The horizontal stress σ_x under the center of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0$].

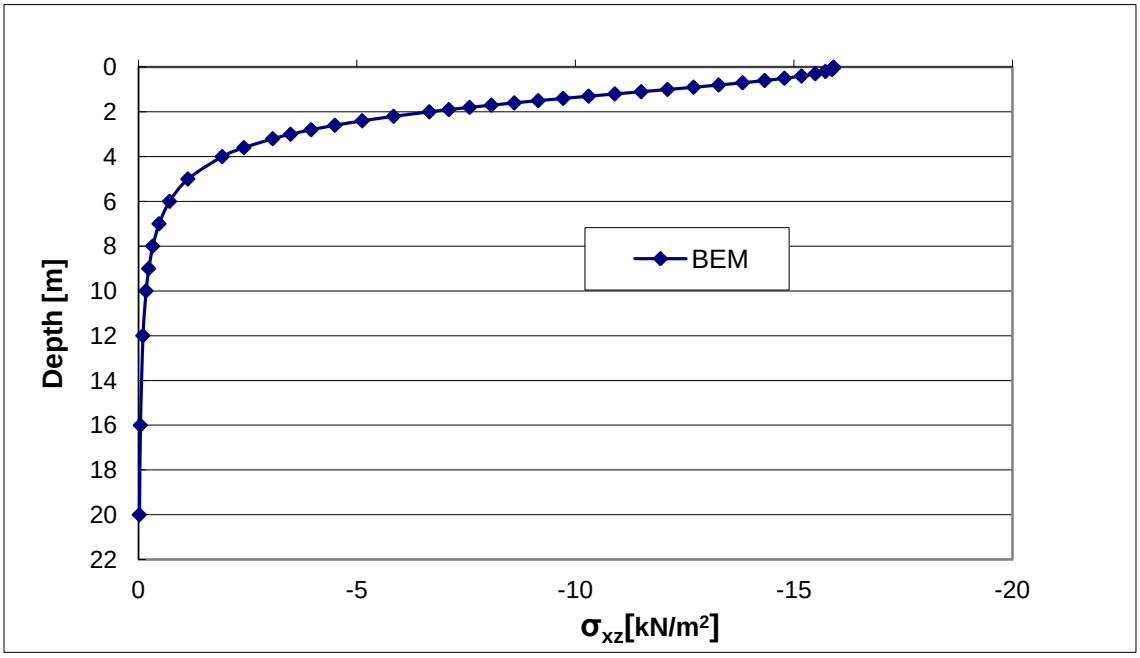


Figure 5.71 The shear stress σ_{xz} under the corner of a uniform square load [Mindlin's BEM solution, $\upsilon = 0.3$].

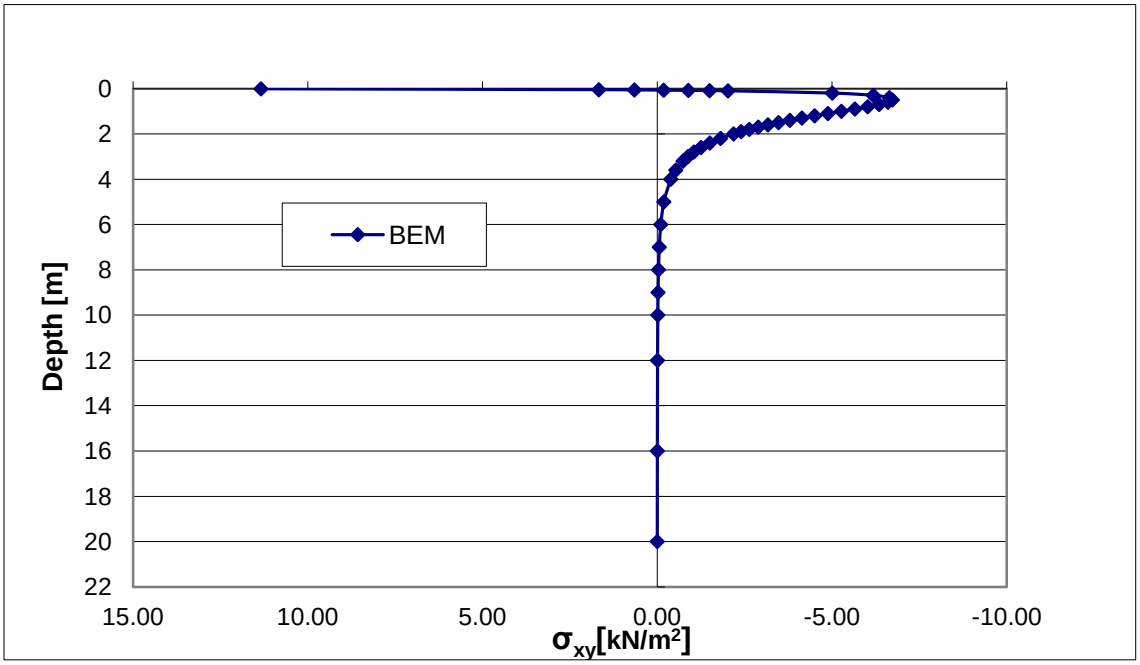
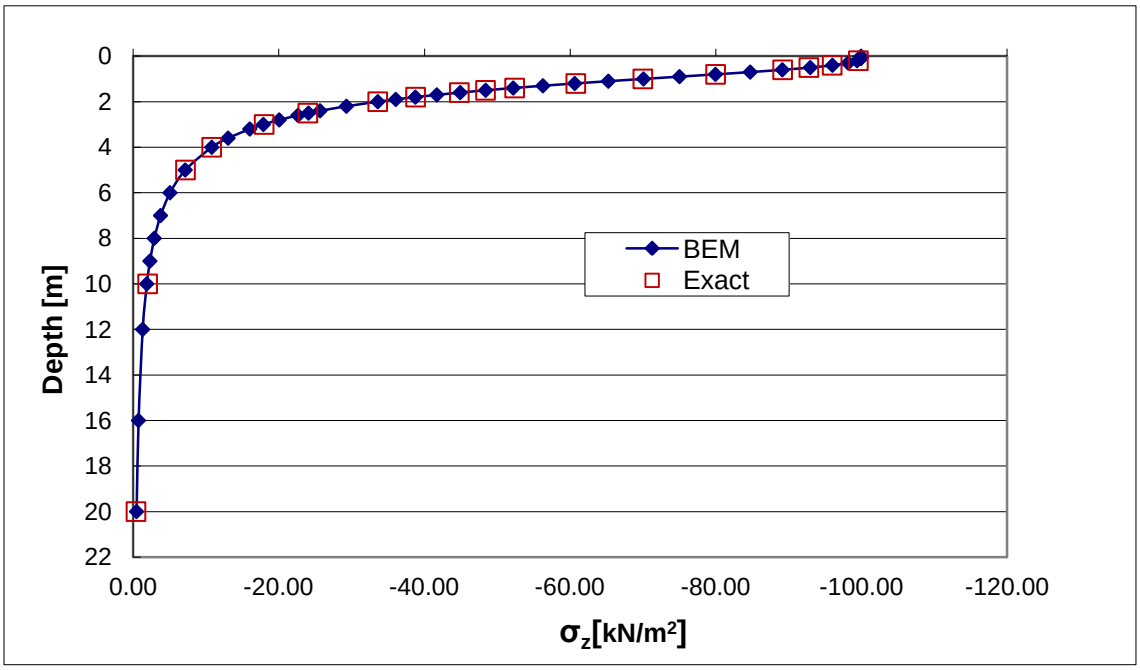


Figure 5.72 The shear stress σ_{xy} under the corner of a uniform square load [Mindlin's BEM solution, $\upsilon = 0.3$].

a.



b.

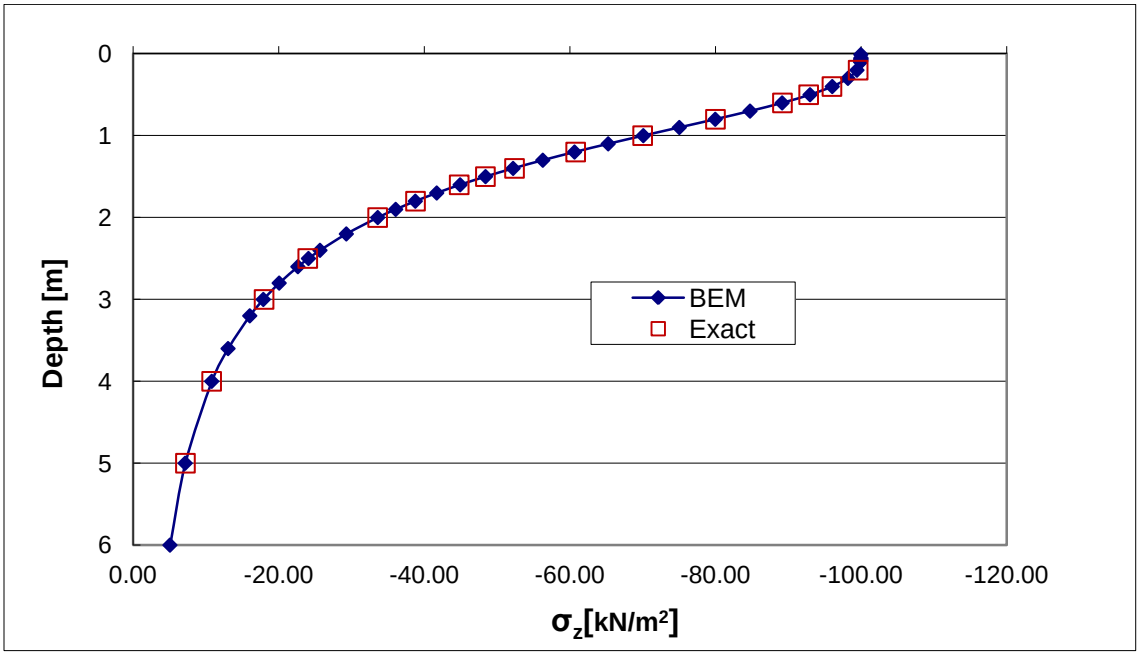
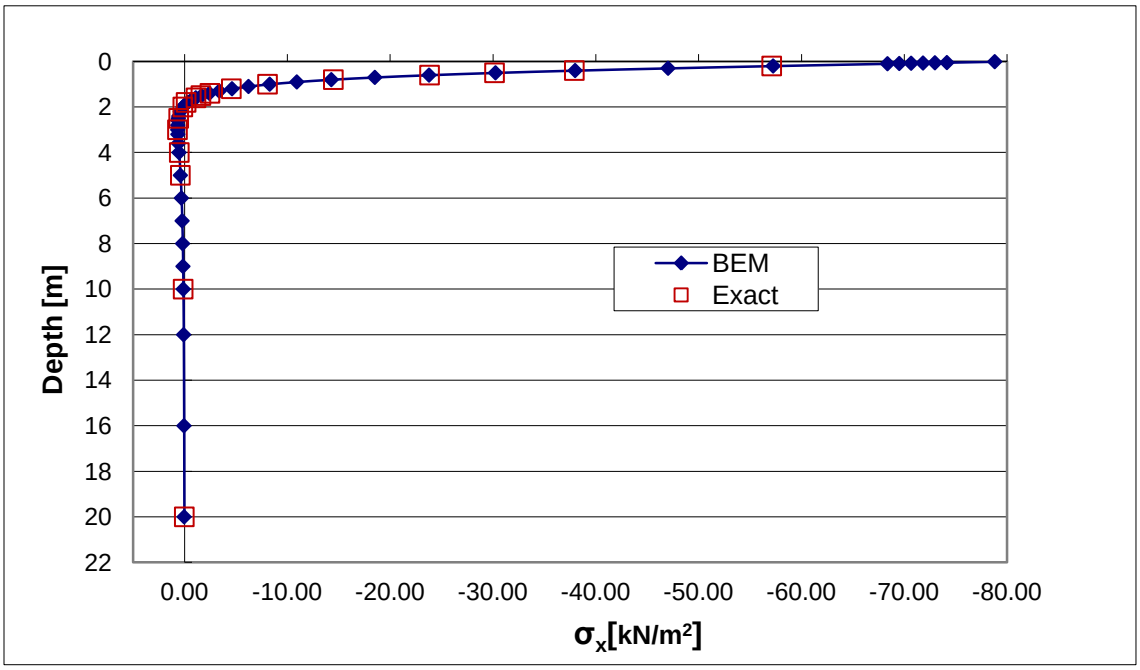


Figure 5.73 a. & b. The vertical stress σ_z under the center of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0.3$].

a.



b.

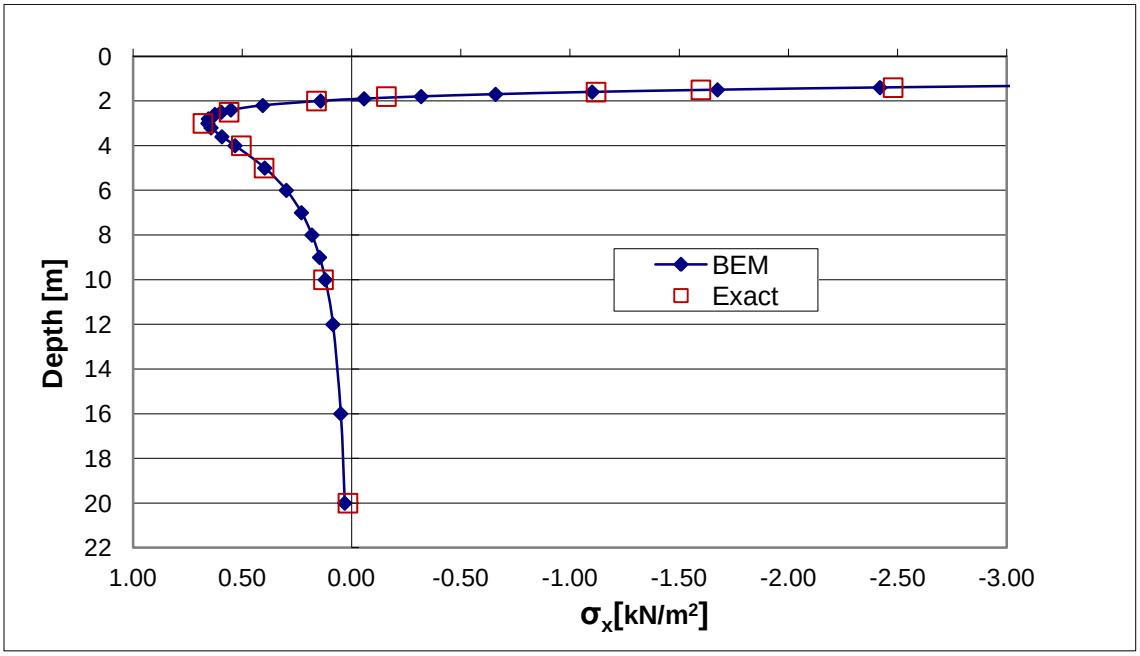


Figure 5.74 a. & b. The horizontal stress σ_x under the center of a uniform square load [Analytical and Mindlin's BEM solutions, $\upsilon = 0.3$].

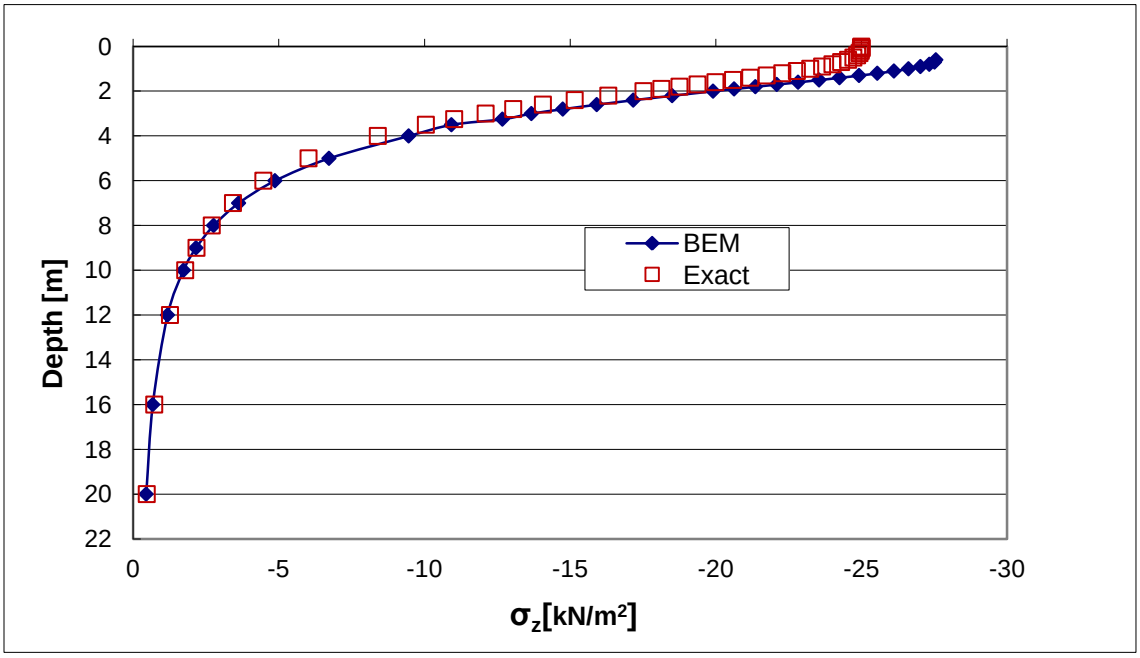
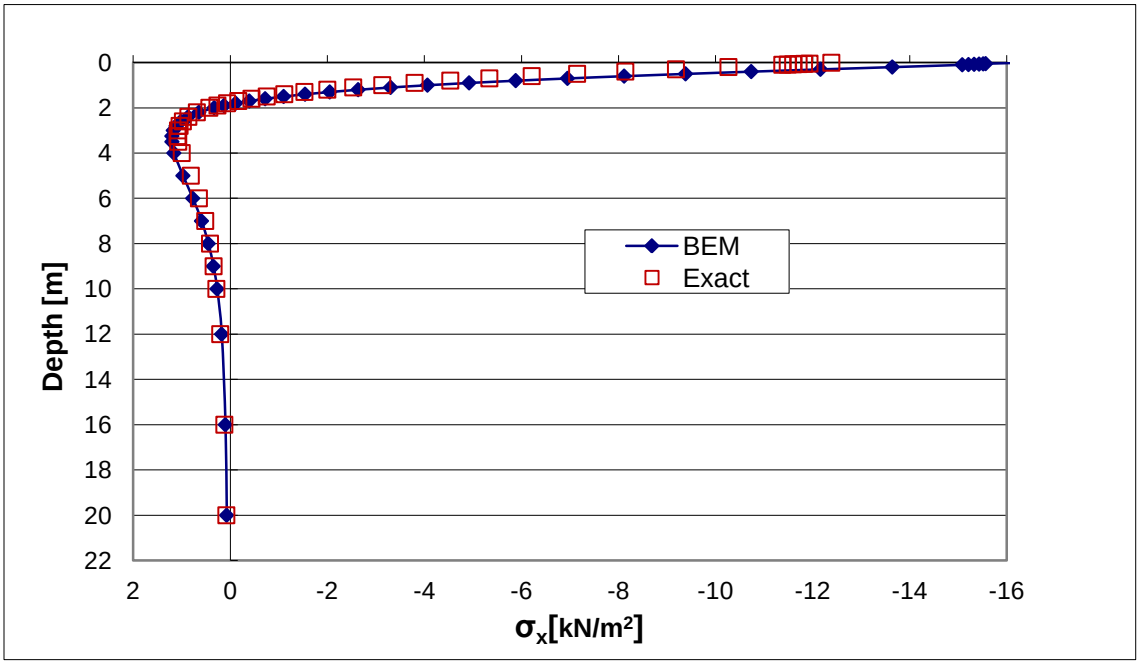


Figure 5.79 The vertical stress σ_z under the corner of a uniform square load [Analytical and Infinite BE solutions, $\upsilon = 0$].

a.



b.

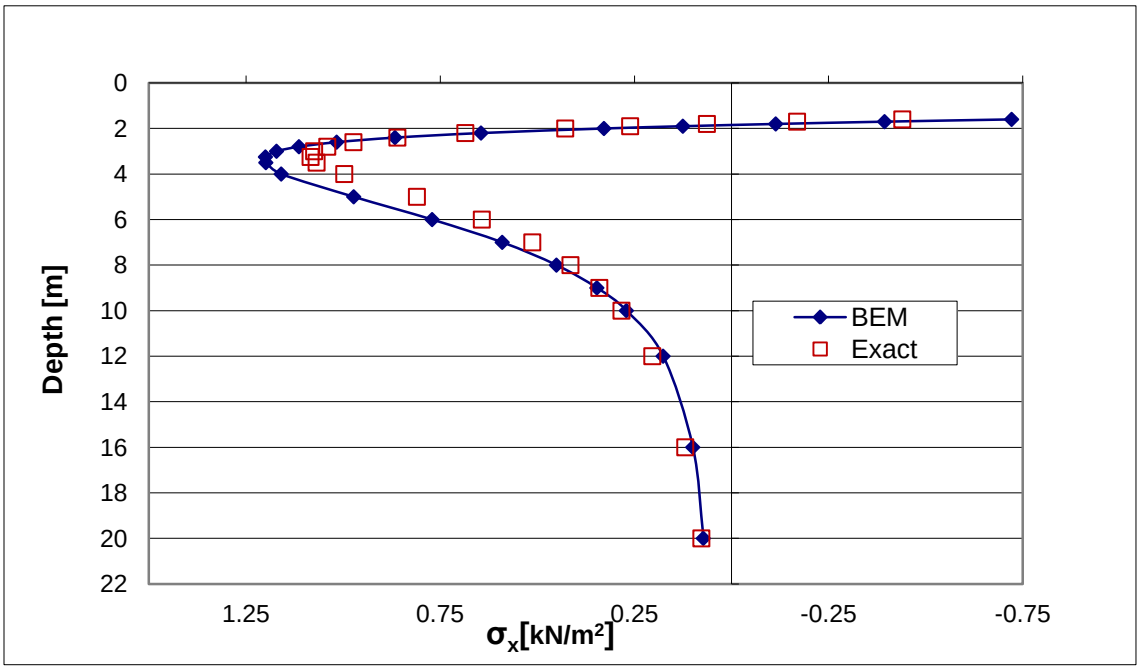


Figure 5.80 a. & b. The horizontal stress σ_x under the corner of a uniform square load [Analytical and Infinite BE solutions, $\upsilon = 0$].

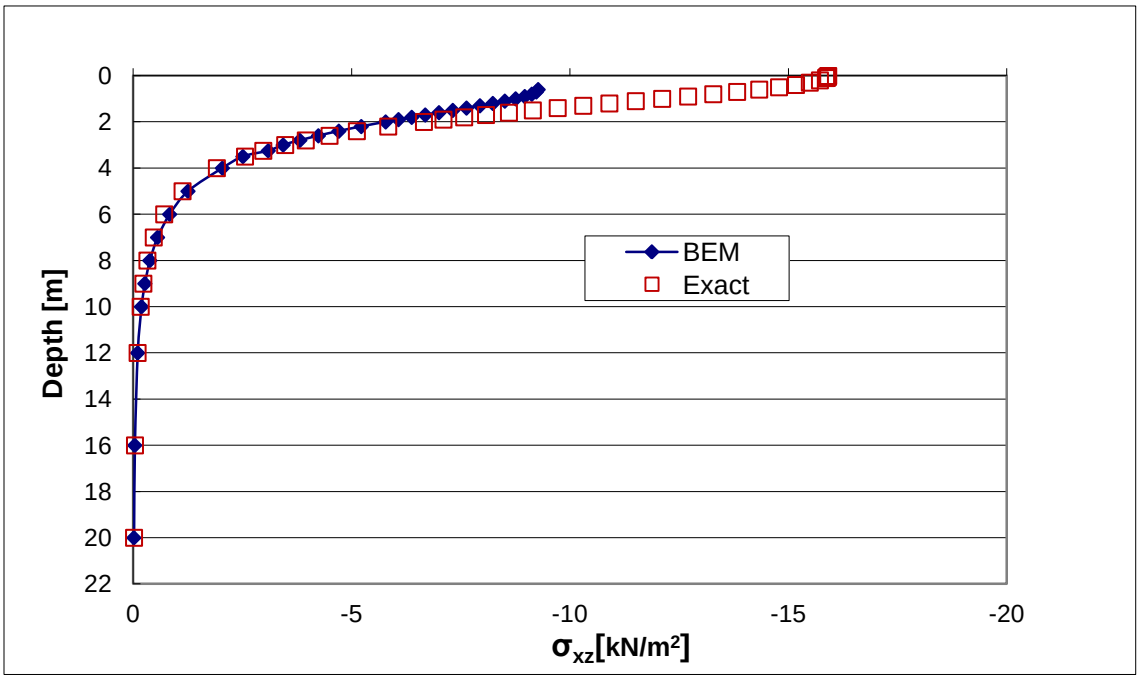


Figure 5.81 The shear stress σ_{xz} under the corner of a uniform square load [Analytical and Infinite BE solutions, $\upsilon = 0$].

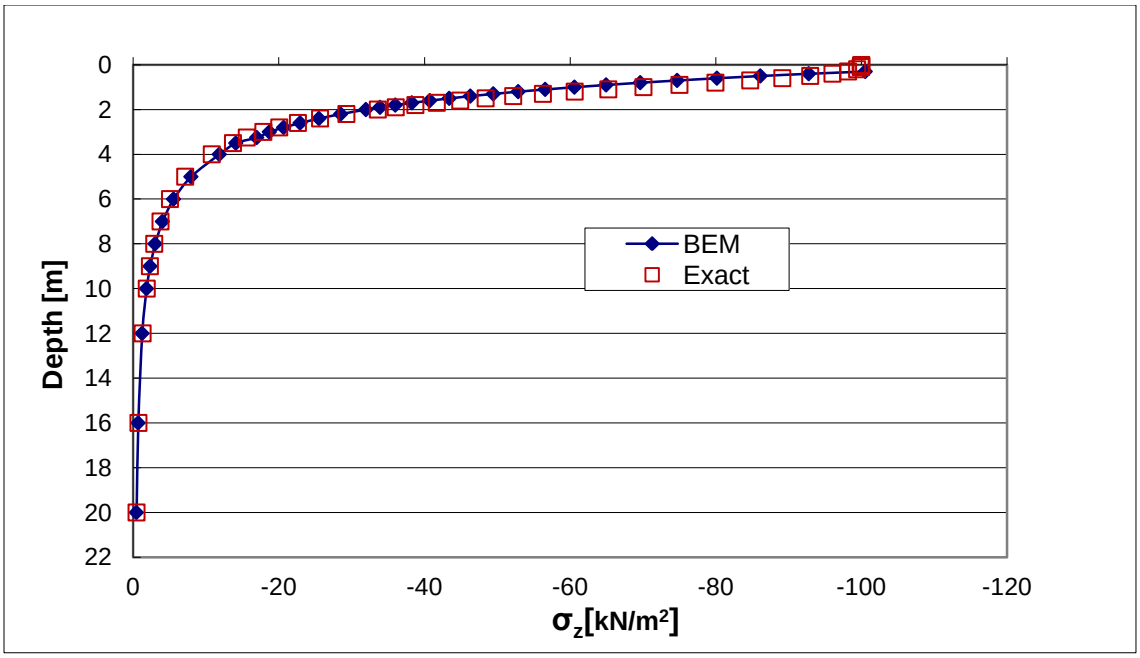


Figure 5.82 The vertical stress σ_z under the center of a uniform square load [Analytical and Infinite BE solutions, $\upsilon = 0$].

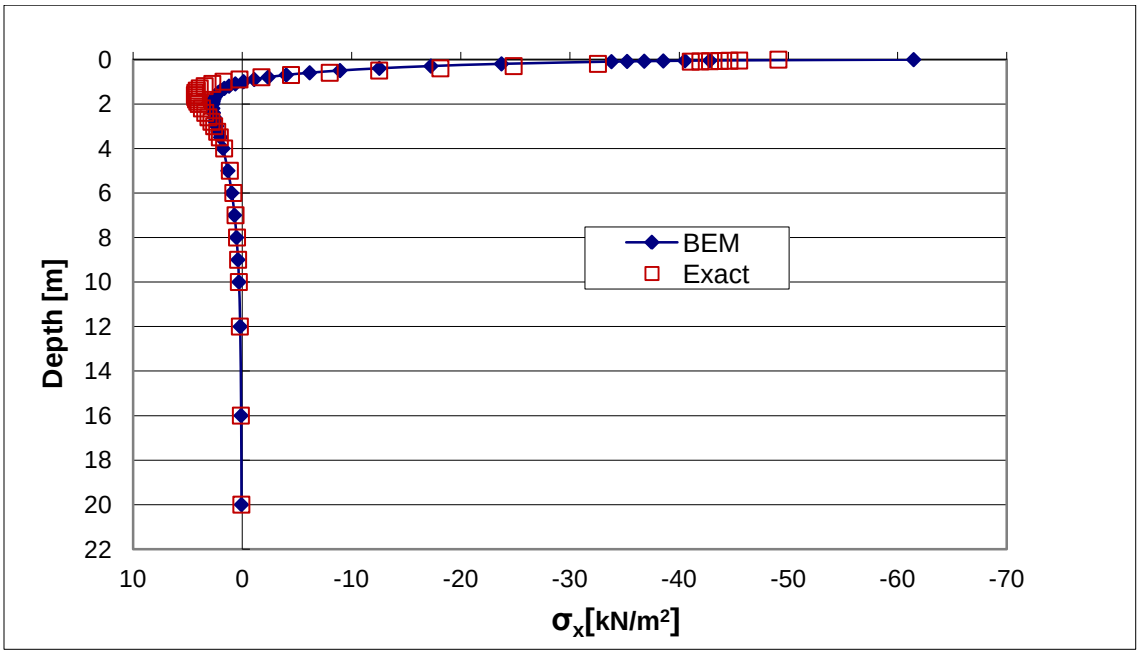


Figure 5.83 The horizontal stress σ_x under the center of a uniform square load [Analytical and Infinite BE solutions, $\upsilon = 0$].

Table 5.21 Uncoupled Flac^{3D} and exact stress under the center of a loaded square area [kN/m2] , ν = 0

	Depth [m]	σ_z Exact	σ_z Numerical	R.E%	σ_x Exact	σ_x Numerical	R.E%
Flac^{3D} sub-domain	0.06	99.98	100.03	0.05	44.39	41.38	-6.78
	0.19	99.53	99.54	0.01	33.60	30.63	-8.82
	0.31	97.97	97.92	-0.05	23.96	21.07	-12.06
	0.44	95.00	94.87	-0.13	15.91	13.12	-17.53
	0.56	90.67	90.53	-0.15	9.59	6.90	-28.08
	0.69	85.32	85.28	-0.05	4.86	2.23	-54.04
	0.81	79.36	79.54	0.22	1.47	-1.14	-177.27
	0.94	73.17	73.70	0.72	-0.86	-3.51	310.15
	1.12	64.16	64.59	0.67	-2.93	-5.79	97.40
	1.37	53.37	54.76	2.61	-4.10	-7.36	79.64
	1.62	44.33	46.77	5.49	-4.33	-8.17	88.38
	1.86	36.99	40.47	9.40	-4.16	-8.64	107.83
	2.11	31.10	35.59	14.44	-3.82	-9.01	135.63
	2.36	26.37	31.83	20.71	-3.45	-9.37	171.52
	2.61	22.55	28.93	28.28	-3.08	-9.74	215.77
	2.85	19.46	26.70	37.20	-2.75	-10.14	268.78

Table 5.22 Uncoupled Flac^{3D} and exact vertical displacement under the center of a loaded square area [m], ν = 0

	Depth [m]	u_z Exact	u_z Numerical	R.E%
Flac^{3D} sub-domain	0.13	-0.0211944	-0.0160478	-24.28
	0.25	-0.0199509	-0.0148032	-25.80
	0.38	-0.0187271	-0.0135779	-27.50
	0.50	-0.0175403	-0.0123893	-29.37
	0.63	-0.0164075	-0.0112537	-31.41
	0.75	-0.0153413	-0.0101827	-33.63
	0.88	-0.0143494	-0.0091829	-36.00
	1.00	-0.0134347	-0.0082561	-38.55
	1.25	-0.0118311	-0.0066415	-43.86
	1.50	-0.0104993	-0.0052716	-49.79
	1.75	-0.0093951	-0.0041006	-56.35
	2.00	-0.0084754	-0.0030865	-63.58
	2.25	-0.0077036	-0.0021941	-71.52
	2.50	-0.0070502	-0.0013957	-80.20
	2.75	-0.0064921	-0.0006698	-89.68
	3.00	-0.0060111	0.0000000	-100.00

Table 5.23 Uncoupled Flac^{3D} and exact vertical displacement under the corner of a loaded square area [m], ν = 0

	Depth [m]	u_z Exact	u_z Numerical	R.E%
Flac^{3D} sub-domain	0.13	-0.0109095	-0.0061713	-43.43
	0.25	-0.0105972	-0.0058230	-45.05
	0.38	-0.0102856	-0.0054867	-46.66
	0.50	-0.0099755	-0.0051566	-48.31
	0.63	-0.0096677	-0.0048300	-50.04
	0.75	-0.0093635	-0.0045059	-51.88
	0.88	-0.0090640	-0.0041843	-53.84
	1.00	-0.0087702	-0.0038657	-55.92

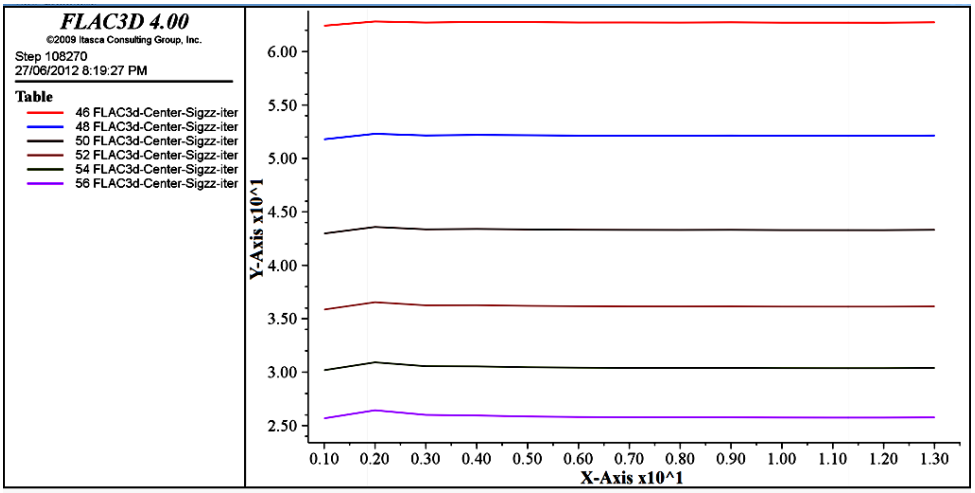


Figure 5.115 Convergence of vertical stress (σ_z) at chosen zones in the Flac3D sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ σ_z).*

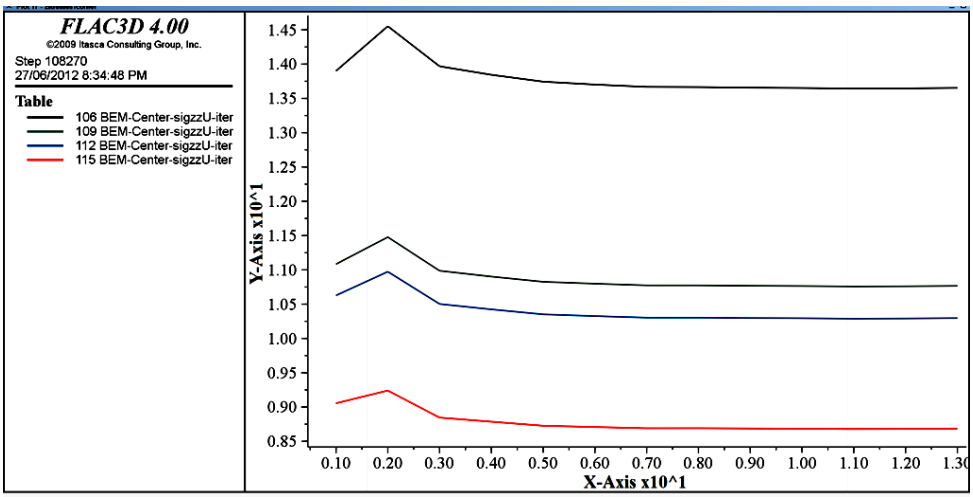


Figure 5.116 Convergence of the vertical stress (σ_z) at chosen zones in the BEM sub-domain over the iterative scheme, *(X-Axis ≡ iterations) and (Y-Axis ≡ σ_z).*

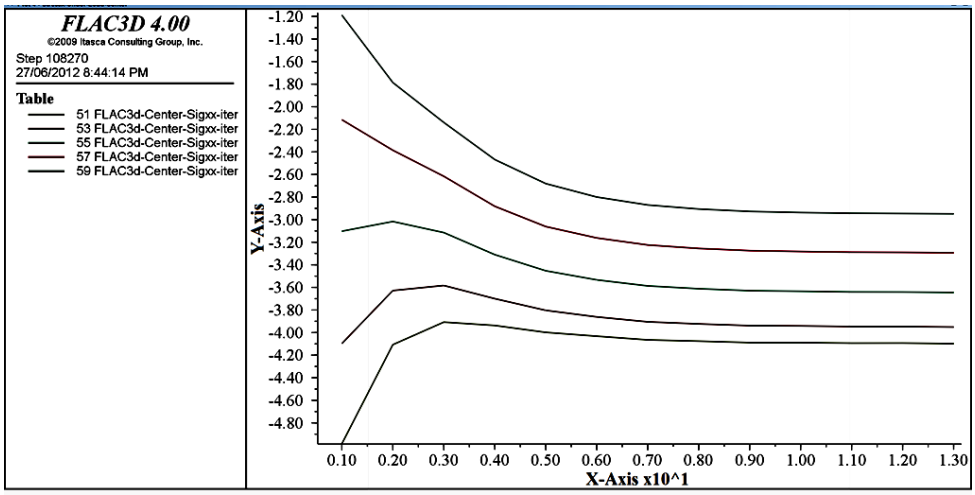


Figure 5.117 Convergence of the horizontal stress (σ_x) at chosen zones in the Flac3D sub-domain over the iterative scheme, *(X-Axis ≡iterations) and (Y-Axis ≡ σ_x).*

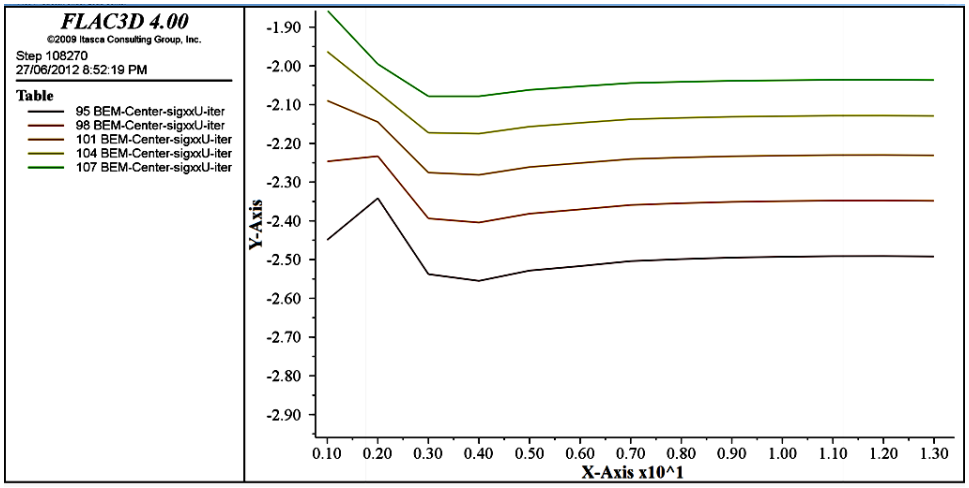


Figure 5.118 Convergence of the horizontal stress (σ_x) at chosen zones in the BEM sub-domain over the iterative scheme, *(X-Axis ≡iterations) and (Y-Axis ≡ σ_x).*

Table 5.29 Coupled (First Method/IDDM) and exact vertical (σ_z) and horizontal (σ_x) stress under the center of a loaded square area [kN/m^2], ν = 0.3

	Depth [m]	Exact σz	Numerical σz	R.E%
Flac^{3D} sub-domain	0.06	99.98	99.99	0.005
	0.19	99.53	99.48	-0.046
	0.31	97.97	97.84	-0.136
	0.44	95.00	94.75	-0.264
	0.56	90.67	90.30	-0.408
	0.69	85.32	84.86	-0.539
	0.81	79.36	78.86	-0.629
	0.94	73.17	72.69	-0.653
	1.12	64.16	62.97	-1.855
	1.37	53.37	52.32	-1.955
	1.62	44.33	43.45	-1.990
	1.86	36.99	36.25	-2.017
	2.11	31.10	30.46	-2.063
	2.36	26.37	25.81	-2.127
	2.61	22.55	22.06	-2.199
	2.85	19.46	19.01	-2.281
BEM sub-domain	3.00	17.89	18.01	0.627
	3.10	16.93	17.21	1.694
	3.20	16.03	16.04	0.068
	3.30	15.20	15.15	-0.368
	3.40	14.43	14.36	-0.483
	3.50	13.72	13.65	-0.495
	4.00	10.81	10.76	-0.451
	4.10	10.33	10.29	-0.440
	4.50	8.71	8.68	-0.400
	5.00	7.16	7.14	-0.361
	5.50	5.98	5.96	-0.331
	6.00	5.07	5.05	-0.308
	6.20	4.76	4.75	-0.301
	6.50	4.35	4.34	-0.291
	7.00	3.77	3.76	-0.277
	7.50	3.30	3.29	-0.266
	8.00	2.91	2.90	-0.257
	10.00	1.88	1.87	-0.236

	Depth [m]	Exact σx	Numerical σx
Flac^{3D} sub-domain	0.06		72.472
	0.19		58.364
	0.20	57.120	
	0.31		45.603
	0.40	37.920	
	0.44		34.729
	0.50	30.160	
	0.56		25.911
	0.60	23.840	
	0.69		19.034
	0.80	14.480	
	0.81		13.823
	0.94		9.954
	1.00	8.080	
	1.12		5.936
	1.20	4.560	
	1.37		2.853
	1.40	2.480	
	1.50	1.600	
	1.60	1.120	
	1.62		1.186
	1.80	0.160	
	1.86		0.292
	2.00	-0.160	
	2.11		-0.177
	2.36		-0.416
	2.50	-0.560	
	2.61		-0.529
	2.85		-0.573
BEM sub-domain	3.00	-0.680	-0.698
	3.10		-0.676
	3.20		-0.661
	3.30		-0.637
	3.40		-0.619
	3.50		-0.604
	4.00	-0.504	-0.529
	4.10		-0.514
	4.50		-0.457
	5.00	-0.400	-0.394
	5.50		-0.341
	6.00		-0.297
	6.20		-0.281
	6.50		-0.260
	7.00		-0.228
	7.50		-0.202
	8.00		-0.180
	10.00	-0.128	-0.120

Table 5.30 Uncoupled and exact vertical (σ_z) stress under the center of a loaded square area [kN/m^2], ν = 0.3

	Depth [m]	Exact σz	Numerical σz	R.E%
Flac^{3D} sub-domain	0.06	99.98	100.00	0.017
	0.19	99.53	99.54	0.009
	0.31	97.97	97.97	-0.001
	0.44	95.00	94.99	-0.009
	0.56	90.67	90.68	0.012
	0.69	85.32	85.41	0.102
	0.81	79.36	79.60	0.303
	0.94	73.17	73.64	0.646
	1.12	64.16	64.32	0.255
	1.37	53.37	54.21	1.588
	1.62	44.33	45.92	3.578
	1.86	36.99	39.31	6.272
	2.11	31.10	34.12	9.723
	2.36	26.37	30.05	13.980
	2.61	22.55	26.85	19.070
	2.85	19.46	24.32	24.990

Table 5.31 Coupled (First Method/IDDM) vertical displacement (u_z) under the center (left) and the corner (right) of a loaded square area[m], ν = 0.3

	Depth [m]	Exact uz	Numerical uz	R.E%
Flac^{3D} sub-domain	0.13	-0.01972	-0.01959	-0.66
	0.25	-0.01892	-0.01879	-0.69
	0.38	-0.01804	-0.01790	-0.73
	0.50	-0.01711	-0.01698	-0.77
	0.63	-0.01618	-0.01604	-0.82
	0.75	-0.01526	-0.01512	-0.86
	0.88	-0.01437	-0.01424	-0.91
	1.00	-0.01353	-0.01340	-0.96
	1.25	-0.01201	-0.01191	-0.82
	1.50	-0.01072	-0.01065	-0.68
	1.75	-0.00963	-0.00958	-0.57
	2.00	-0.00871	-0.00867	-0.47
	2.25	-0.00794	-0.00790	-0.40
	2.50	-0.00728	-0.00725	-0.33
	2.75	-0.00671	-0.00669	-0.28
	3.00	-0.00622	-0.00620	-0.23
BEM sub-domain	3.10	-0.00604	-0.00602	-0.28
	3.20	-0.00587	-0.00585	-0.31
	3.30	-0.00571	-0.00569	-0.32
	3.40	-0.00556	-0.00554	-0.31
	3.50	-0.00541	-0.00540	-0.31
	4.00	-0.00479	-0.00477	-0.28
	4.10	-0.00468	-0.00467	-0.28
	4.50	-0.00429	-0.00428	-0.27
	5.00	-0.00388	-0.00387	-0.25
	5.50	-0.00354	-0.00353	-0.24
	6.00	-0.00326	-0.00325	-0.24
	6.20	-0.00315	-0.00315	-0.23
	6.50	-0.00301	-0.00301	-0.23
	7.00	-0.00280	-0.00280	-0.23
	7.50	-0.00262	-0.00261	-0.22
	8.00	-0.00246	-0.00245	-0.22
	10.00	-0.00197	-0.00197	-0.21

	Depth [m]	Exact uz	Numerical uz	R.E%
Flac^{3D} sub-domain	0.13	-0.01004	-0.01013	0.91
	0.25	-0.00986	-0.00992	0.57
	0.38	-0.00967	-0.00970	0.36
	0.50	-0.00946	-0.00948	0.22
	0.63	-0.00924	-0.00925	0.12
	0.75	-0.00902	-0.00902	0.04
	0.88	-0.00879	-0.00879	-0.02
	1.00	-0.00856	-0.00855	-0.07
BEM sub-domain	3.1	-0.00524	-0.00523	-0.26
	3.2	-0.00513	-0.00512	-0.26
	3.3	-0.00502	-0.00501	-0.26
	3.4	-0.00492	-0.00490	-0.26
	3.5	-0.00482	-0.00480	-0.26
	4	-0.00436	-0.00435	-0.25
	4.1	-0.00427	-0.00426	-0.25
	4.5	-0.00397	-0.00396	-0.24
	5	-0.00364	-0.00363	-0.24
	5.5	-0.00335	-0.00335	-0.23
	6	-0.00311	-0.00310	-0.23
	6.5	-0.00289	-0.00289	-0.23
	7	-0.00271	-0.00270	-0.22
	7.5	-0.00254	-0.00254	-0.22
	8	-0.00239	-0.00239	-0.22
	10	-0.00194	-0.00194	-0.21

Table 5.32 Uncoupled Flac^{3D} vertical displacement (u_z) under the center (left) and the corner (right) of a loaded square area[m], ν = 0.3

	Depth [m]	Exact uz	Numerical uz	R.E%
Flac^{3D} sub-domain	0.13	-0.02119	-0.01605	-24.28
	0.25	-0.01995	-0.01480	-25.80
	0.38	-0.01873	-0.01358	-27.50
	0.50	-0.01754	-0.01239	-29.37
	0.63	-0.01641	-0.01125	-31.41
	0.75	-0.01534	-0.01018	-33.63
	0.88	-0.01435	-0.00918	-36.00
	1.00	-0.01344	-0.00826	-38.55
	1.25	-0.01183	-0.00664	-43.86
	1.50	-0.01050	-0.00527	-49.79
	1.75	-0.00940	-0.00410	-56.35
	2.00	-0.00848	-0.00309	-63.58
	2.25	-0.00770	-0.00219	-71.52
	2.50	-0.00705	-0.00140	-80.20
	2.75	-0.00649	-0.00067	-89.68
	3.00	-0.00601	0.00000	-100.00

	Depth [m]	Exact uz	Numerical uz	R.E%
Flac^{3D} sub-domain	0.13	-0.01091	-0.00617	-43.43
	0.25	-0.01060	-0.00582	-45.05
	0.38	-0.01029	-0.00549	-46.66
	0.50	-0.00998	-0.00516	-48.31
	0.63	-0.00967	-0.00483	-50.04
	0.75	-0.00936	-0.00451	-51.88
	0.88	-0.00906	-0.00418	-53.84
	1.00	-0.00877	-0.00387	-55.92

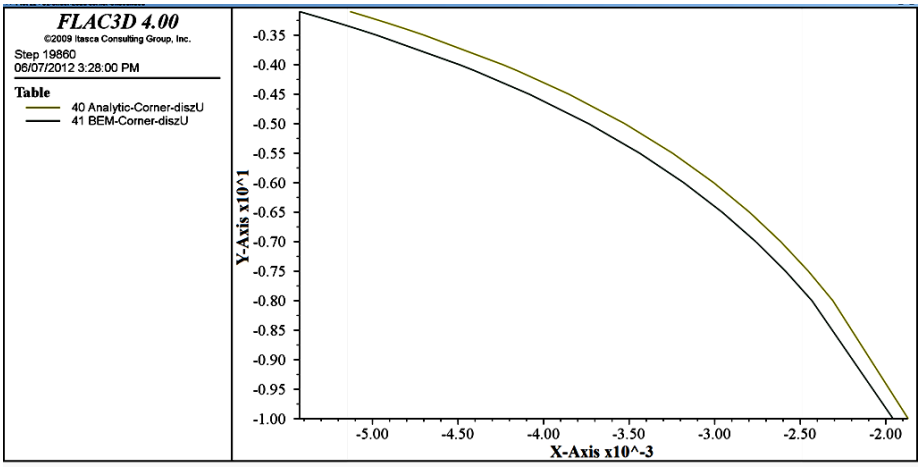
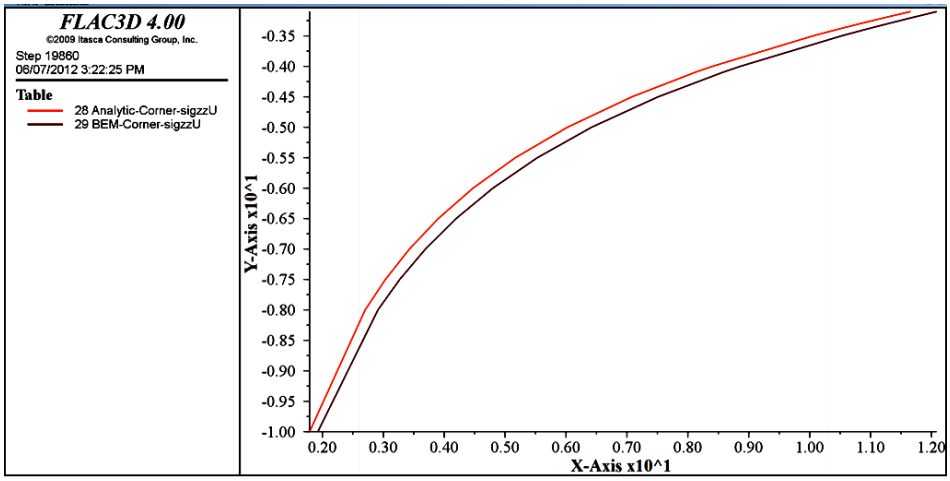
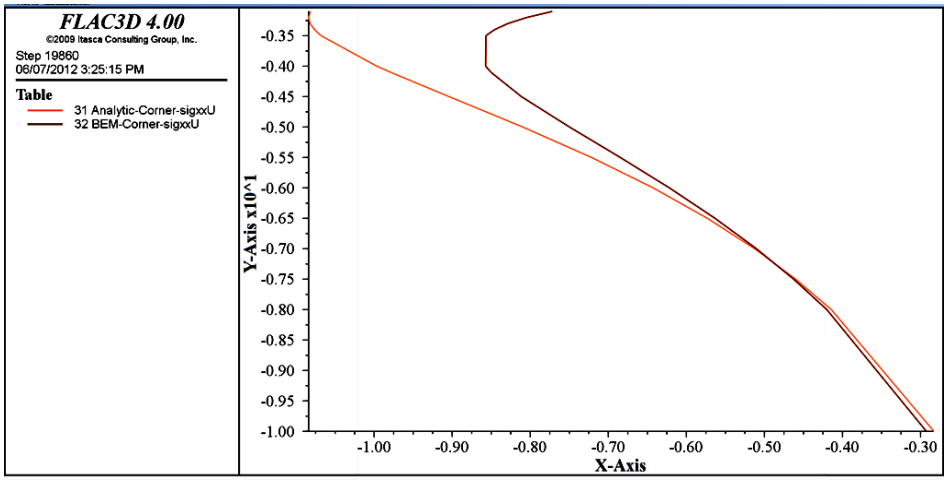


Figure 5.143 Coupled (Second Method/SDDM) and exact vertical displacement (u_z) solution under the corner of a uniform square load in BEM sub-domain, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ u_Z).*

a.



b.



c.

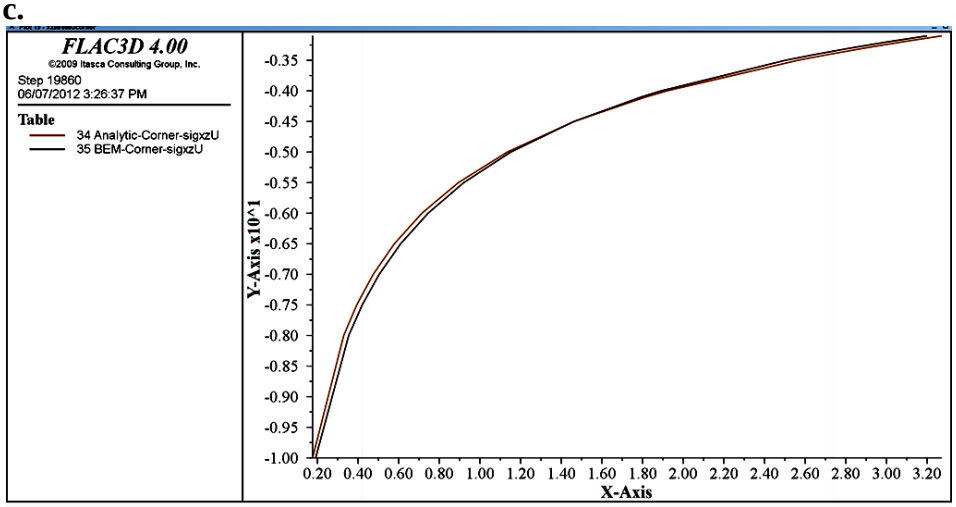


Figure 5.144 Coupled (Second Method/SDDM) and exact stress components {a. σ_z, b. σ_x and c. σ_{xz}} under the center of a uniform square load in BEM sub-domains, ν = 0.0, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

Table 5.34 Coupled (Second Method/SDDM) and the exact vertical (σ_z) and horizontal (σ_x) stress under the center of a loaded square area [kN/m^2], ν = 0

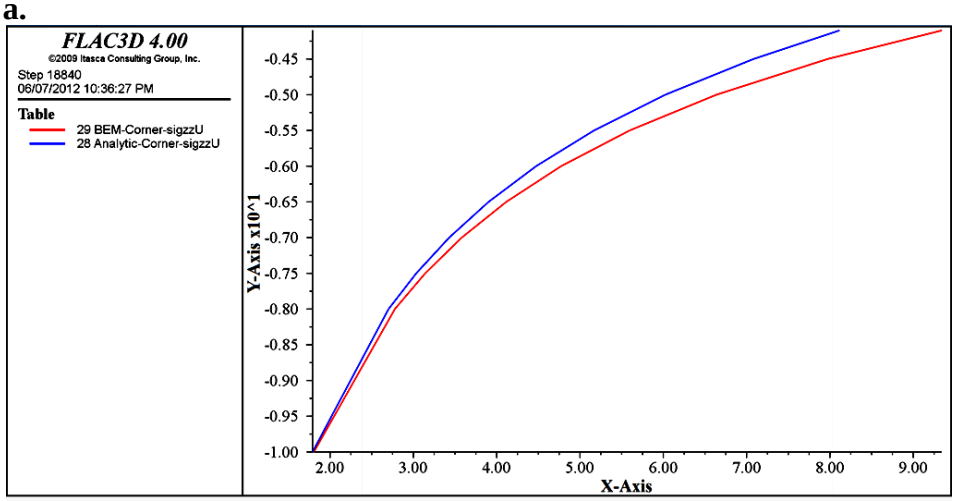
	Depth [m]	Exact σz	Numerical σz	R.E%	Exact σx	Numerical σx	R.E%
Flac^{3D} sub-domain	0.06	99.98	100.01	0.03	44.39	46.62	5.02
	0.19	99.53	99.49	-0.04	33.60	35.74	6.37
	0.31	97.97	97.80	-0.17	23.96	26.06	8.76
	0.44	95.00	94.67	-0.34	15.91	18.03	13.28
	0.56	90.67	90.21	-0.51	9.59	11.74	22.44
	0.69	85.32	84.81	-0.60	4.86	7.05	44.97
	0.81	79.36	78.91	-0.57	1.47	3.67	148.78
	0.94	73.17	72.87	-0.40	-0.86	1.31	-252.92
	1.12	64.16	63.40	-1.18	-2.93	-0.87	-70.47
	1.37	53.37	53.06	-0.58	-4.10	-2.25	-45.00
	1.62	44.33	44.49	0.35	-4.33	-2.79	-35.73
	1.86	36.99	37.56	1.53	-4.16	-2.92	-29.77
	2.11	31.10	32.01	2.93	-3.82	-2.89	-24.54
	2.36	26.37	27.56	4.51	-3.45	-2.79	-19.12
	2.61	22.55	23.97	6.28	-3.08	-2.68	-13.08
	2.85	19.46	21.05	8.20	-2.75	-2.58	-6.20
BEM sub-domain	3.00	17.89	18.34	2.51	-2.57	-2.65	2.99
	3.10	16.93	17.42	2.94	-2.45	-2.45	-0.22
	3.20	16.03	16.28	1.54	-2.34	-2.25	-4.19
	3.30	15.20	15.39	1.20	-2.24	-2.09	-6.75
	3.40	14.43	14.61	1.19	-2.14	-1.96	-8.39
	3.50	13.72	13.90	1.33	-2.05	-1.86	-9.44
	4.00	10.81	11.08	2.55	-1.66	-1.48	-10.76
	4.10	10.33	10.63	2.83	-1.59	-1.42	-10.58
	4.50	8.71	9.05	3.92	-1.36	-1.23	-9.26
	5.00	7.16	7.53	5.09	-1.13	-1.05	-7.07
	5.10	6.90	7.27	5.29	-1.09	-1.02	-6.63
	5.50	5.98	6.34	6.00	-0.95	-0.91	-4.92
	6.00	5.07	5.41	6.67	-0.81	-0.79	-3.03
	6.20	4.76	5.09	6.88	-0.77	-0.75	-2.36
	6.50	4.35	4.66	7.14	-0.70	-0.69	-1.44
	7.00	3.77	4.05	7.46	-0.61	-0.61	-0.14
	7.50	3.30	3.55	7.66	-0.54	-0.54	0.91
	8.00	2.91	3.13	7.77	-0.47	-0.48	1.76
	10.00	1.88	2.02	7.68	-0.31	-0.32	3.76

Table 5.35 Coupled (Second Method/SDDM) vertical displacement (u_z) under the center (left) and the corner (right) of a loaded square area[m], ν = 0.0

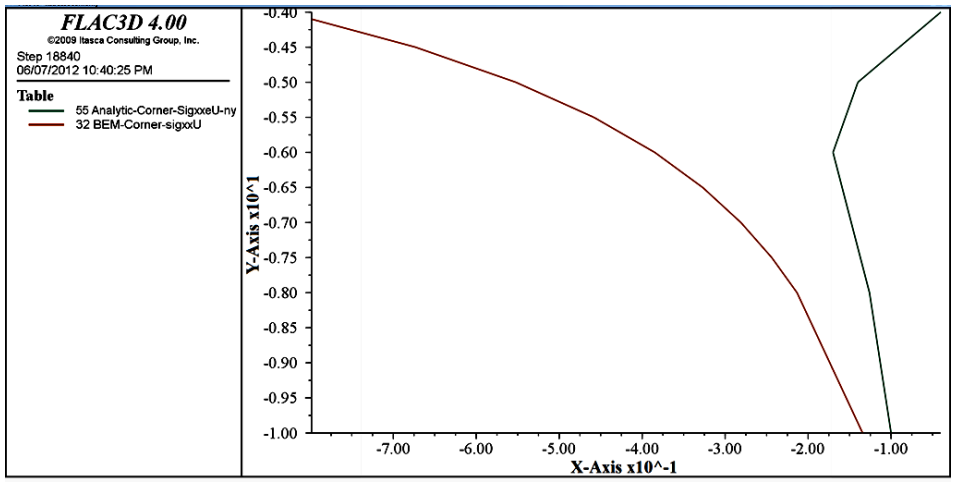
	Depth [m]	Exact uz	Numerical uz	R.E%
Flac^{3D} sub-domain	0.13	-0.02119	-0.02164	2.10
	0.25	-0.01995	-0.02040	2.23
	0.38	-0.01873	-0.01917	2.38
	0.50	-0.01754	-0.01799	2.55
	0.63	-0.01641	-0.01686	2.73
	0.75	-0.01534	-0.01579	2.92
	0.88	-0.01435	-0.01480	3.13
	1.00	-0.01344	-0.01388	3.33
	1.25	-0.01183	-0.01230	3.94
	1.50	-0.01050	-0.01097	4.48
	1.75	-0.00940	-0.00986	4.91
	2.00	-0.00848	-0.00892	5.20
	2.25	-0.00770	-0.00811	5.33
	2.50	-0.00705	-0.00742	5.30
	2.75	-0.00649	-0.00682	5.09
	3.00	-0.00601	-0.00630	4.72
BEM sub-domain	3.10	-0.00584	-0.00611	4.68
	3.20	-0.00567	-0.00594	4.76
	3.30	-0.00552	-0.00578	4.86
	3.40	-0.00537	-0.00563	4.96
	3.50	-0.00523	-0.00549	5.06
	4.00	-0.00462	-0.00487	5.48
	4.10	-0.00451	-0.00476	5.54
	4.50	-0.00413	-0.00437	5.74
	5.00	-0.00374	-0.00396	5.88
	5.10	-0.00367	-0.00388	5.89
	5.50	-0.00341	-0.00361	5.91
	6.00	-0.00314	-0.00332	5.87
	6.20	-0.00304	-0.00321	5.84
	6.50	-0.00290	-0.00307	5.78
	7.00	-0.00270	-0.00285	5.67
	7.50	-0.00252	-0.00266	5.54
	8.00	-0.00237	-0.00250	5.39
	10.00	-0.00190	-0.00199	4.81

	Depth [m]	Exact uz	Numerical uz	R.E%
Flac^{3D} sub-domain	0.13	-0.01091	-0.01163	6.60
	0.25	-0.01060	-0.01128	6.46
	0.38	-0.010290	-0.01095	6.43
	0.50	-0.009980	-0.01062	6.45
	0.63	-0.00967	-0.01030	6.50
	0.75	-0.00936	-0.00998	6.57
	0.88	-0.00906	-0.00967	6.64
	1.00	-0.00877	-0.00936	6.71
BEM sub-domain	3.1	-0.00513	-0.00543	5.76
	3.2	-0.00502	-0.00531	5.80
	3.3	-0.00491	-0.00520	5.84
	3.4	-0.00480	-0.00509	5.87
	3.5	-0.00470	-0.00498	5.90
	4	-0.00424	-0.00450	5.99
	4.1	-0.00416	-0.00441	6.00
	4.5	-0.00385	-0.00408	6.02
	5	-0.00353	-0.00374	5.99
	5.5	-0.00325	-0.00344	5.92
	6	-0.00301	-0.00318	5.82
	6.5	-0.00280	-0.00296	5.71
	7	-0.00261	-0.00276	5.57
	7.5	-0.00245	-0.00259	5.43
	8	-0.00231	-0.00243	5.29
	10	-0.00187	-0.00196	4.72

a.



b.



c.

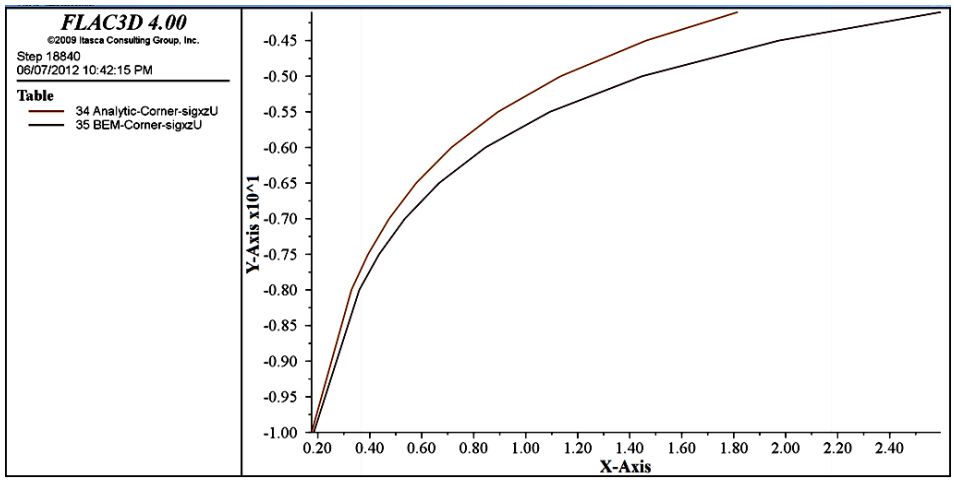


Figure 5.150 Coupled (Second Method/SDDCM) and exact stress components {a. σ_z, b. σ_x and c. σ_{xz}} under the center of a uniform square load in BEM sub-domain, ν = 0.3, *(Y-Axis ≡ Depth) and (X-Axis ≡ σ_x).*

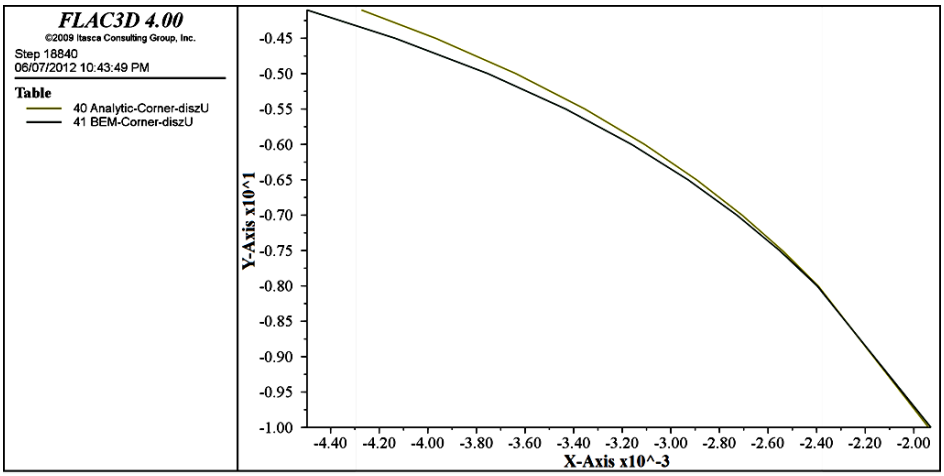


Figure 5.151 Coupled (Second Method/SDDM) and exact vertical displacement (u_z) solution under the corner of a uniform square load in BEM sub-domain, ν = 0.3, *(-Axis ≡ Depth) and (X-Axis ≡ u_Z).*

Table 5.37 Coupled (Second Method/SDDM) and the exact vertical (σ_z) and horizontal (σ_x) stress under the center of a loaded square area [kN/m^2], ν = 0.3

	Depth [m]	Exact σz	Numerical σz	R.E%
Flac^{3D} sub-domain	0.06	99.98	99.99	0.008
	0.19	99.53	99.47	-0.053
	0.31	97.97	97.79	-0.180
	0.44	95.00	94.65	-0.363
	0.56	90.67	90.15	-0.571
	0.69	85.32	84.67	-0.761
	0.81	79.36	78.65	-0.898
	0.94	73.17	72.49	-0.935
	1.19	61.30	58.84	-4.005
	1.56	46.40	44.49	-4.123
	1.93	35.37	33.80	-4.429
	2.30	27.42	26.00	-5.187
	2.67	21.69	20.29	-6.452
	3.04	17.48	16.05	-8.227
	3.41	14.34	12.83	-10.541
	3.78	11.95	10.34	-13.449
BEM sub-domain	4.00	10.81	13.53	25.224
	4.10	10.33	13.06	26.324
	4.50	8.71	10.22	17.348
	5.00	7.16	8.10	13.172
	5.10	6.90	7.76	12.514
	5.50	5.98	6.60	10.272
	6.00	5.07	5.48	8.121
	6.20	4.76	5.11	7.410
	6.50	4.35	4.63	6.469
	7.00	3.77	3.96	5.168
	7.50	3.30	3.43	4.121
	8.00	2.91	3.00	3.267
	10.00	1.88	1.90	1.048

	Depth [m]	Exact σx	Numerical σx
Flac^{3D} sub-domain	-0.06		72.951
	-0.19		58.655
	-0.20	57.120	
	-0.31		45.746
	-0.40	37.920	
	-0.44		34.764
	-0.50	30.160	
	-0.56		25.872
	-0.60	23.840	
	-0.69		18.944
	-0.80	14.480	
	-0.81		13.692
	-0.94		9.783
	-1.00	8.080	
	-1.19		4.483
	-1.20	4.560	
	-1.40	2.480	
	-1.56		0.630
	-1.50	1.600	
	-1.60	1.120	
	-1.80	0.160	
	-1.93		-1.145
	-2.00	-0.160	
	-2.30		-2.043
	-2.50	-0.560	
	-2.67		-2.572
	-3.00	-0.680	
	-3.04		-2.947
	-3.41		-3.258
	-3.78		-3.544
BEM sub-domain	4.00	-0.504	-2.033
	4.10		-1.870
	4.50		-1.372
	5.10		-0.987
	5.00	-0.400	-0.930
	5.50		-0.745
	6.00		-0.582
	6.20		-0.532
	6.50		-0.467
	7.00		-0.384
	7.50		-0.321
	8.00		-0.272
	10.00	-0.128	-0.158

Table 5.38 Coupled (Second Method/SDDM) vertical displacement (u_z) under the center (left) and the corner (right) of a loaded square area[m], ν = 0.3

	Depth [m]	Exact uz	Numerical uz	R.E %
Flac^{3D} sub-domain	0.13	-0.01972	-0.01997	1.24
	0.25	-0.01892	-0.01916	1.30
	0.38	-0.01804	-0.01828	1.37
	0.50	-0.01711	-0.01736	1.45
	0.63	-0.01618	-0.01643	1.54
	0.75	-0.01525	-0.01551	1.65
	0.88	-0.01437	-0.01462	1.77
	1.00	-0.01353	-0.01378	1.90
	1.38	-0.01134	-0.01168	2.99
	1.75	-0.00963	-0.01002	4.05
	2.13	-0.00831	-0.00872	4.99
	2.50	-0.00728	-0.00770	5.82
	2.88	-0.00645	-0.00688	6.53
	3.25	-0.00579	-0.00620	7.15
	3.63	-0.00524	-0.00565	7.68
	4.00	-0.00479	-0.00518	8.10
BEM sub-domain	-4.10	-0.00468	-0.00503	7.40
	-4.50	-0.00429	-0.00453	5.62
	-5.00	-0.00388	-0.00404	4.09
	-5.10	-0.00381	-0.00396	3.85
	-5.50	-0.00354	-0.00365	2.99
	-6.00	-0.00326	-0.00333	2.16
	-6.20	-0.00315	-0.00321	1.89
	-6.50	-0.00301	-0.00306	1.53
	-7.00	-0.00280	-0.00283	1.03
	-7.50	-0.00262	-0.00264	0.64
	-8.00	-0.00246	-0.00247	0.32
	-10.00	-0.00197	-0.00197	-0.46

	Depth [m]	Exact uz	Numerical uz	R.E %
Flac^{3D} sub-domain	0.13	-0.01004	-0.01058	5.37
	0.25	-0.00986	-0.01036	5.11
	0.38	-0.00967	-0.01014	4.92
	0.50	-0.00946	-0.00991	4.81
	0.63	-0.00924	-0.00968	4.76
	0.75	-0.00902	-0.00945	4.76
	0.88	-0.00879	-0.00921	4.80
	1.00	-0.00856	-0.00897	4.85
BEM sub-domain	4.1	-0.00427	-0.00450	5.24
	4.5	-0.00397	-0.00414	4.20
	5	-0.00364	-0.00375	3.17
	5.5	-0.00335	-0.00343	2.36
	6	-0.00311	-0.00316	1.72
	6.5	-0.00289	-0.00293	1.21
	7	-0.00271	-0.00273	0.80
	7.5	-0.00254	-0.00255	0.46
	8	-0.00239	-0.00240	0.19
	10	-0.00194	-0.00193	-0.51

4. Appendix d

Table 5.40 Uncoupled Flac^{3D} normalized radial and tangential stress in an infinite medium in the vicinity of a cylindrical tunnel

	r	Exact $[\sigma_r/p]$	Numerical $[\sigma_r/p]$	R.E. % in $[\sigma_r]$	Exact $[\sigma_\theta/p]$	Numerical $[\sigma_\theta/p]$	R.E. % in $[\sigma_\theta]$
FLAC^{3D} Sub-domain	1.03	0.0587	0.0804	36.96	1.9413	2.5861	33.22
	1.09	0.1632	0.2195	34.49	1.8368	2.4469	33.21
	1.16	0.2512	0.3367	34.02	1.7488	2.3296	33.21
	1.22	0.3260	0.4363	33.81	1.6740	2.2300	33.21
	1.28	0.3902	0.5216	33.69	1.6098	2.1445	33.21
	1.34	0.4456	0.5954	33.62	1.5544	2.0707	33.22
	1.41	0.4938	0.6596	33.57	1.5062	2.0066	33.22
	1.47	0.5359	0.7157	33.54	1.4641	1.9505	33.22
	1.53	0.5731	0.7651	33.51	1.4269	1.9011	33.23
	1.59	0.6059	0.8088	33.49	1.3941	1.8573	33.23
	1.66	0.6351	0.8476	33.47	1.3649	1.8185	33.23
	1.72	0.6611	0.8823	33.45	1.3389	1.7839	33.24
	1.78	0.6845	0.9134	33.44	1.3155	1.7528	33.24
	1.84	0.7055	0.9414	33.43	1.2945	1.7248	33.24
	1.91	0.7245	0.9667	33.42	1.2755	1.6995	33.24
	1.97	0.7417	0.9896	33.41	1.2583	1.6766	33.24

Table 5.41 Uncoupled Flac^{3D} radial displacement in an infinite medium in the vicinity of a cylindrical tunnel

	r	Exact u_r [m]	Numerical u_r [m]	R.E. %
FLAC^{3D} Sub-domain	1.00	-0.001250	-0.001873	49.81
	1.06	-0.001176	-0.001788	51.96
	1.13	-0.001111	-0.001714	54.25
	1.19	-0.001053	-0.001649	56.66
	1.25	-0.001000	-0.001592	59.20
	1.31	-0.000952	-0.001542	61.87
	1.38	-0.000909	-0.001497	64.67
	1.44	-0.000870	-0.001457	67.60
	1.50	-0.000833	-0.001422	70.66
	1.56	-0.000800	-0.001391	73.85
	1.63	-0.000769	-0.001363	77.17
	1.69	-0.000741	-0.001338	80.62
	1.75	-0.000714	-0.001316	84.20
	1.81	-0.000690	-0.001296	87.91
	1.88	-0.000667	-0.001278	91.75
	1.94	-0.000645	-0.001263	95.73
	2.00	-0.000625	-0.001249	99.83

Table 5.42 Coupled (First Method/IDDM) normalized radial and tangential stress in an infinite medium in the vicinity of a cylindrical tunnel

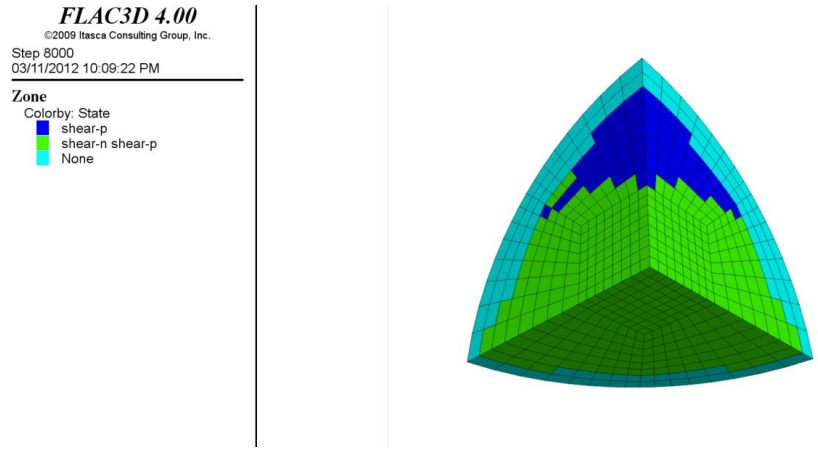
	r	Exact [σ_r/p]	Numerical [σ_r/p]	R.E. % in [σ_r]	Exact [σ_θ/p]	Numerical [σ_θ/p]	R.E. % in [σ_θ]
FLAC^{3D} Sub-domain	1.03	0.0587	0.0609	3.86	1.9413	1.9308	-0.54
	1.09	0.1632	0.1654	1.33	1.8368	1.8260	-0.59
	1.16	0.2512	0.2534	0.87	1.7488	1.7388	-0.57
	1.22	0.3260	0.3287	0.80	1.6740	1.6648	-0.55
	1.28	0.3902	0.3928	0.66	1.6098	1.6011	-0.54
	1.34	0.4456	0.4483	0.60	1.5544	1.5462	-0.53
	1.41	0.4938	0.4966	0.57	1.5062	1.4989	-0.49
	1.47	0.5359	0.5391	0.58	1.4641	1.4575	-0.45
	1.53	0.5731	0.5765	0.59	1.4269	1.4209	-0.42
	1.59	0.6059	0.6095	0.60	1.3941	1.3878	-0.46
	1.66	0.6351	0.6390	0.62	1.3649	1.3584	-0.48
	1.72	0.6611	0.6657	0.69	1.3389	1.3328	-0.46
	1.78	0.6845	0.6898	0.78	1.3155	1.3103	-0.39
	1.84	0.7055	0.7109	0.76	1.2945	1.2899	-0.35
	1.91	0.7245	0.7304	0.82	1.2755	1.2714	-0.32
	1.97	0.7417	0.7487	0.94	1.2583	1.2548	-0.28
BEM Sub-domain	2.00	0.7500	0.7526	0.35	1.2500	1.2439	-0.49
	2.05	0.7620	0.7639	0.24	1.2380	1.2325	-0.44
	2.10	0.7732	0.7757	0.32	1.2268	1.2212	-0.45
	2.25	0.8025	0.8044	0.25	1.1975	1.1929	-0.38
	2.50	0.8400	0.8414	0.17	1.1600	1.1567	-0.29
	2.75	0.8678	0.8689	0.14	1.1322	1.1297	-0.22
	3.00	0.8889	0.8899	0.11	1.1111	1.1091	-0.18
	3.10	0.8959	0.8969	0.11	1.1041	1.1022	-0.17
	3.25	0.9053	0.9062	0.10	1.0947	1.0930	-0.15
	3.50	0.9184	0.9192	0.09	1.0816	1.0802	-0.13
	4.00	0.9375	0.9382	0.07	1.0625	1.0614	-0.10
	4.50	0.9506	0.9512	0.06	1.0494	1.0485	-0.08
	5.00	0.9600	0.9605	0.05	1.0400	1.0393	-0.07
	6.00	0.9722	0.9726	0.04	1.0278	1.0273	-0.05
	8.00	0.9844	0.9846	0.02	1.0156	1.0154	-0.03
	10.00	0.9900	0.9901	0.01	1.0100	1.0098	-0.02

Table 5.43 Coupled (First Method/IDDM) radial displacement in an infinite medium in the vicinity of a cylindrical tunnel

	r	Exact u_r [m]	Numerical u_r [m]	R.E. % in [ur]
FLAC^{3D} Sub-domain	1.00	-0.001250	-0.001236	-1.12
	1.06	-0.001176	-0.001163	-1.14
	1.13	-0.001111	-0.001098	-1.18
	1.19	-0.001053	-0.001040	-1.21
	1.25	-0.001000	-0.000988	-1.25
	1.31	-0.000952	-0.000940	-1.28
	1.38	-0.000909	-0.000897	-1.30
	1.44	-0.000870	-0.000858	-1.33
	1.50	-0.000833	-0.000822	-1.35
	1.56	-0.000800	-0.000789	-1.38
	1.63	-0.000769	-0.000758	-1.40
	1.69	-0.000741	-0.000730	-1.41
	1.75	-0.000714	-0.000704	-1.43
	1.81	-0.000690	-0.000680	-1.45
	1.88	-0.000667	-0.000657	-1.48
	1.94	-0.000645	-0.000636	-1.48
BEM Sub-domain	2.00	-0.000625	-0.000616	-1.49
	2.05	-0.000610	-0.000602	-1.32
	2.10	-0.000595	-0.000587	-1.32
	2.25	-0.000556	-0.000548	-1.32
	2.50	-0.000500	-0.000493	-1.32
	2.75	-0.000455	-0.000448	-1.34
	3.00	-0.000417	-0.000411	-1.36
	3.10	-0.000403	-0.000398	-1.36
	3.25	-0.000385	-0.000379	-1.37
	3.50	-0.000357	-0.000352	-1.39
	4.00	-0.000313	-0.000308	-1.41
	4.50	-0.000278	-0.000274	-1.43
	5.00	-0.000250	-0.000246	-1.45
	6.00	-0.000208	-0.000205	-1.47
	8.00	-0.000156	-0.000154	-1.49
	10.00	-0.000125	-0.000123	-1.50

5. Appendix e

a.



b.

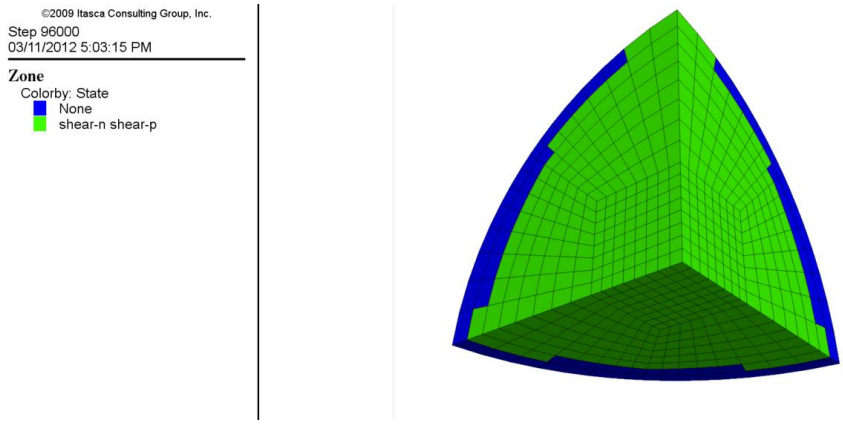
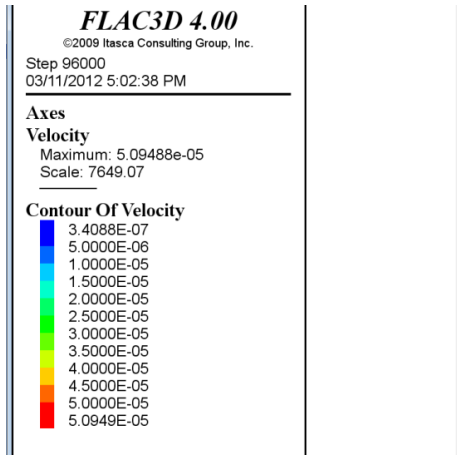
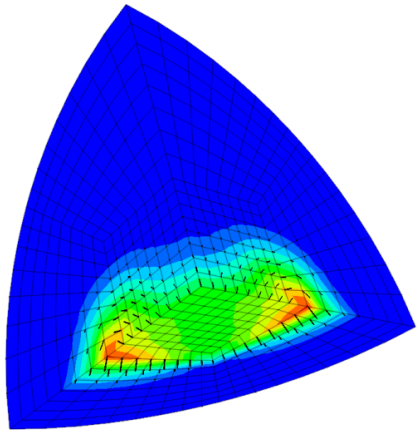


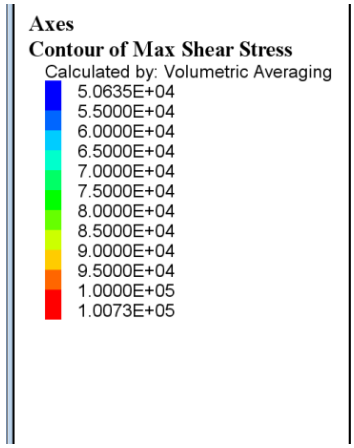
Figure 5.184 Yield zone in a. The uncoupled solution, b. The coupled solution, L/a = 2.6.

a.





b.



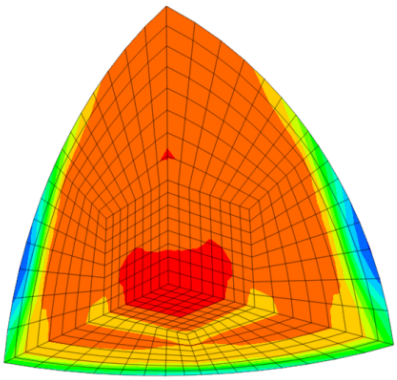


Figure 5.185 Contour plots after applying the coupled solution for: a. The velocity field, b. The maximum shear stress field.

6. Appendix f

```
void FishExample::UMelan(double **UMelan,double *dxr,double *dxrr,
                                                    double rr,double cc,double xdash,double E,double ny)
{
//-------------------------------------------
//   FUNDAMENTAL SOLUTION FOR DISPLACEMENTS
//   isotropic material (Melan solution)
//-------------------------------------------
// Cd         :  Cartesian dimension (2D)
// dxr[size=cd]   :  rx,ry.
// dxrr[cd]   :  Rx,Ry.
// r          :  r.
// rr         :  R.
// cc         :  c.
// xdash      :  y-.
// E          :  Young's modulus.
// ny         :  Poisson's ratio.
// UMelan[cd][cd]: Melan displacement solution.

const double Pi=3.14159265359;
double G= E/(2.0*(1+ny));
double Kd= 1.0/(8.0*Pi*G*(1.0 - ny));
double theta= atan(dxrr[1]/dxrr[0]);
UMelan[0][0] =Kd*(-(8*pow((1-ny),2)-(3-4*ny))*(log(rr))+((3-4*ny)*pow(dxrr[0],2)-
2*cc*xdash)/pow(rr,2)+4*cc*xdash*(pow(dxrr[0],2))/pow(rr,4));
UMelan[0][1] =Kd*(((3-4*ny)*dxr[0]*dxr[1]/pow(rr,2))+(4*cc*xdash*dxrr[0]*dxr[1]/pow(rr,4))-(4*(1-ny)*(1-
2*ny)*theta));
UMelan[1][0] =Kd*(((3-4*ny)*dxr[0]*dxr[1]/pow(rr,2))-(4*cc*xdash*dxrr[0]*dxr[1]/pow(rr,4))+(4*(1-ny)*(1-
2*ny)*theta));
UMelan[1][1] =Kd*(-(8*(pow((1-ny),2))-(3-4*ny))*(log(rr))+((3-4*ny)*pow(dxr[1],2)+2*cc*xdash)/pow(rr,2)-
4*cc*xdash*(pow(dxr[1],2))/pow(rr,4));
return;
}
```

```
void  FishExample::TMelan(double **TKMelan,double *dxr,double *Vnor,double *dxrr,double rr,double cc,
                                                                                    double xdash,double
ny)
 {
//-------------------------------------------
//   FUNDAMENTAL SOLUTION FOR TRACTIONS
//   isotropic material (Melan solution)
//-------------------------------------------
// Cd            :    Cartesian dimension (2D)

// dxr[size=cd]   :   rx,ry.
// dxrr[cd]       :   Rx, Ry.
// rr             :   R.
// Vnor[cd]       :   normal vector.
// ny             :   Poisson's ratio.
// cc             :   c.
// xdash          :   y-
// TMelan[Cd][Cd] :   Melan traction solution.
double ***Sigma;    // Temporary array.
Sigma= new double**[Cd];
  for (int i = 0; i < Cd; ++i)
  {
    Sigma[i] = new double*[Cd];
    for (int j = 0; j < Cd; ++j)
      Sigma[i][j] = new double[Cd];
  }
double Pi=3.14159265359;
double Ks=1/(4*Pi*(1-ny));
Sigma[0][0][0]=-Ks*(((3*xdash+cc)*(1-2*ny))/pow(rr,2)+(2*dxrr[0]*(pow(dxrr[0],2)+2*cc*xdash)
4*xdash*pow(dxr[1],2)*(1-2*ny))/pow(rr,4)-(16*cc*xdash*dxrr[0]*pow(dxr[1],2))/pow(rr,6));
Sigma[0][1][0]=-Ks*dxr[1]*(-(1-2*ny)/pow(rr,2)+2*(pow(xdash,2)-2*cc*xdash-pow(cc,2)+2*xdash*dxrr[0]*(1-
2*ny))/pow(rr,4)+(16*cc*xdash*pow(dxrr[0],2))/pow(rr,6));
Sigma[1][1][0]=-Ks*((xdash+3*cc)*(1-2*ny)/pow(rr,2)+2*(dxrr[0]*(pow(dxr[1],2)+2*pow(cc,2))-
2*cc*pow(dxr[1],2)+2*xdash*pow(dxr[1],2)*(1-
2*ny))/pow(rr,4)+(16*cc*xdash*dxrr[0]*pow(dxr[1],2))/pow(rr,6));
Sigma[0][0][1]=-Ks*dxr[1]*((1-2*ny)/pow(rr,2)-2*(pow(cc,2)-pow(xdash,2)+6*cc*xdash-2*xdash*dxrr[0]*(1-
2*ny))/pow(rr,4)+(16*cc*xdash*pow(dxr[1],2))/pow(rr,6));
Sigma[0][1][1]=-Ks*((3*xdash+cc)*(1-2*ny)/pow(rr,2)+2*((2*cc*xdash+pow(dxr[1],2))*dxrr[0]-
2*xdash*pow(dxrr[0],2)*(1-2*ny))/pow(rr,4)-(16*cc*xdash*dxrr[0]*pow(dxr[1],2))/pow(rr,6));
Sigma[1][1][1]=-Ks*dxr[1]*(3*(1-2*ny)/pow(rr,2)+2*(pow(dxr[1],2)-4*cc*xdash-2*pow(cc,2)-2*xdash*dxrr[0]*(1-
2*ny))/pow(rr,4)+(16*cc*xdash*pow(dxrr[0],2))/pow(rr,6));
```

```
Sigma[1][0][0]=Sigma[0][1][0];
Sigma[1][0][1]=Sigma[0][1][1];
for (int i = 0; i < Cd; ++i)
 for (int j = 0; j < Cd; ++j)
    {
        TMelan[i][j]= 0;
    }
for (int i = 0; i < Cd; ++i)
 for (int j = 0; j < Cd; ++j)
  for (int k = 0; k < Cd; ++k)
    {
        TMelan[i][j]= TMelan[i][j]+Sigma[j][k][i]*Vnor[k];
    }
return;
}

void FishExample::UMindlin(double **UMindlin,double *dxr,double *dxrr,double rr,double cc,double xdash,
                                                                                    double E,double
ny)
{
//--------------------------------------------
//   FUNDAMENTAL SOLUTION FOR DISPLACEMENTS
//   isotropic material (Mindlin solution)
//--------------------------------------------
// Cd              :  Cartesian dimension (3D )
// dxr[size=cd]    :  rx,ry,rz.
// dxrr[cd]        :  Rx,Ry,Rz.
// rr              :  R.
// cc              :  c.
// xdash           :  y-
// E               :  Young's modulus.
// ny              :  Poisson's ratio.
// UMindlin[Cd][Cd]:  Mindlin displacement solution
double Pi=3.14159265359;
double G= E/(2.0*(1+ny));
double Kd= 1.0/(16.0*Pi*G*(1.0 - ny));
UMindlin[0][0] =Kd*((8*pow((1-ny),2)-3+4*ny)/rr+((3-4*ny)*pow(dxrr[0],2)-
2*cc*xdash)/pow(rr,3)+6*cc*xdash*pow(dxrr[0],2)/pow(rr,5));
UMindlin[0][1] =Kd*dxr[1]*((3-4*ny)*dxr[0]/pow(rr,3)-(4-4*ny)*(1-
2*ny)/rr/(rr+dxrr[0])+6*cc*xdash*dxrr[0]/pow(rr,5));
```

```
UMindlin[0][2] =dxr[2]*Kd*((3-4*ny)*dxr[0]/pow(rr,3)-(4-4*ny)*(1-
2*ny)/rr/(rr+dxrr[0])+6*cc*xdash*dxrr[0]/pow(rr,5));
UMindlin[1][0] =Kd*dxr[1]*((3-4*ny)*dxr[0]/pow(rr,3)+(4-4*ny)*(1-2*ny)/rr/(rr+dxrr[0])-
6*cc*xdash*dxrr[0]/pow(rr,5));
UMindlin[1][1] =Kd*(1/rr+(3-4*ny)*pow(dxr[1],2)/pow(rr,3)+2*cc*xdash/pow(rr,3)*(1-
3*pow(dxr[1],2)/pow(rr,2))+(4-4*ny)*(1-2*ny)/(rr+dxrr[0])*(1-pow(dxr[1],2)/rr/(rr+dxrr[0])));
UMindlin[1][2] =Kd*dxr[1]*dxr[2]*((3-4*ny)/pow(rr,3)-(4-4*ny)*(1-2*ny)/rr/pow((rr+dxrr[0]),2)-
6*cc*xdash/pow(rr,5));
UMindlin[2][0] =dxr[2]*Kd*((3-4*ny)*dxr[0]/pow(rr,3)+(4-4*ny)*(1-2*ny)/rr/(rr+dxrr[0])-
6*cc*xdash*dxrr[0]/pow(rr,5));
UMindlin[2][1] =Kd*dxr[1]*dxr[2]*((3-4*ny)/pow(rr,3)-(4-4*ny)*(1-2*ny)/rr/pow((rr+dxrr[0]),2)-
6*cc*xdash/pow(rr,5));
UMindlin[2][2] =Kd*(1/rr+(3-4*ny)*pow(dxr[2],2)/pow(rr,3)+2*cc*xdash/pow(rr,3)*(1-
3*pow(dxr[2],2)/pow(rr,2))+(4-4*ny)*(1-2*ny)/(rr+dxrr[0])*(1-pow(dxr[2],2)/rr/(rr+dxrr[0])));

return;
}

void FishExample::TMindlin(double **TMindlin,double *dxr,double *Vnor,double *dxrr,
                                                    double rr,double cc,double xdash,double ny,int
Cd)
{
//--------------------------------------------
//   FUNDAMENTAL SOLUTION FOR TRACTIONS
//   isotropic material (Mindlin solution)
//--------------------------------------------
// Cd              :   Cartesian dimension (3D ).
// dxr[size=cd]    :   rx,ry,rz.
// dxrr[cd]        :   Rx,Ry,Rz.
// rr              :   R.
// Vnor[cd]        :   normal vector.
// ny              :   Poisson's ratio.
// cc              :   c.
// xdash           :   y-
// TMindlin[Cd][Cd]:   Mindlin traction solution.
double ***sigma;  //   Temporary array.
sigma= new double**[Cd];
  for (int i = 0; i < Cd; ++i)
  {
    sigma[i] = new double*[Cd];
    for (int j = 0; j < Cd; ++j)
```

```
      sigma[i][j] = new double[Cd];
  }
double Pi=3.14159265359;
double Ks=1/(8*Pi*(1-ny));
sigma[0][0][0]=Ks*((1-2*ny)*dxr[0]/pow(rr,3)-((9-12*ny)*xdash*pow(dxrr[0],2)-3*cc*dxrr[0]*(5*xdash-
cc))/pow(rr,5)-30*cc*xdash*pow(dxrr[0],3)/pow(rr,7));
sigma[0][1][0]=Ks*dxr[1]*((1-2*ny)/pow(rr,3)-((9-12*ny)*xdash*dxrr[0]-3*cc*(3*xdash+cc))/pow(rr,5)-
30*cc*xdash*pow(dxrr[0],2)/pow(rr,7));
sigma[0][2][0]=Ks*dxr[2]*((1-2*ny)/pow(rr,3)-((9-12*ny)*xdash*dxrr[0]-3*cc*(3*xdash+cc))/pow(rr,5)-
30*cc*xdash*pow(dxrr[0],2)/pow(rr,7));
sigma[1][1][0]=Ks*((1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr,3)-((9-12*ny)*pow(dxr[1],2)*dxr[0]-
6*cc*dxrr[0]*((1-2*ny)*xdash-2*ny*cc))/pow(rr,5)-30*cc*pow(dxr[1],2)*xdash*dxrr[0]/pow(rr,7)-(4-4*ny)*(1-
2*ny)/rr/(rr+dxrr[0])*(1-pow(dxr[1],2)/rr/(rr+dxrr[0])-pow(dxr[1],2)/pow(rr,2)));
sigma[1][2][0]=Ks*dxr[1]*dxr[2]*(-(9-12*ny)*dxr[0]/pow(rr,5)+(4-4*ny)*(1-
2*ny)/pow(rr,2)/(rr+dxrr[0])*(1/(rr+dxrr[0])+1/rr)-30*cc*xdash*dxrr[0]/pow(rr,7));
sigma[2][2][0]=Ks*((1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr,3)-((9-12*ny)*pow(dxr[2],2)*dxr[0]-
6*cc*dxrr[0]*((1-2*ny)*xdash-2*ny*cc))/pow(rr,5)-30*cc*pow(dxr[2],2)*xdash*dxrr[0]/pow(rr,7)-(4-4*ny)*(1-
2*ny)/rr/(rr+dxrr[0])*(1-pow(dxr[2],2)/rr/(rr+dxrr[0])-pow(dxr[2],2)/pow(rr,2)));
sigma[0][0][1]=Ks*dxr[1]*(-(1-2*ny)/pow(rr,3)-(9-12*ny)*pow(dxrr[0],2)/pow(rr,5)+6*cc/pow(rr,5)*(cc+(1-
2*ny)*dxrr[0]+5*xdash*pow(dxrr[0],2)/pow(rr,2)));
sigma[0][1][1]=Ks*((1-2*ny)*dxr[0]/pow(rr,3)-(9-12*ny)*pow(dxr[1],2)*dxrr[0]/pow(rr,5)-
6*cc/pow(rr,5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[1],2)-5*pow(dxr[1],2)*xdash*dxrr[0]/pow(rr,2)));
sigma[0][2][1]=Ks*dxr[1]*dxr[2]*(-(9-12*ny)*dxrr[0]/pow(rr,5)+6*cc/pow(rr,5)*(1-
2*ny+5*xdash*dxrr[0]/pow(rr,2)));
sigma[1][1][1]=Ks*dxr[1]*((1-2*ny)*(5-4*ny)/pow(rr,3)-(9-12*ny)*pow(dxr[1],2)/pow(rr,5)-(4-4*ny)*(1-
2*ny)/rr/pow((rr+dxrr[0]),2)*(3-pow(dxr[1],2)*(3*rr+dxrr[0])/pow(rr,2)/(rr+dxrr[0]))+6*cc/pow(rr,5)*(3*cc-
(3-2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/pow(rr,2)));
sigma[1][2][1]=Ks*dxr[2]*((1-2*ny)/pow(rr,3)-(9-12*ny)*pow(dxr[1],2)/pow(rr,5)-(4-4*ny)*(1-
2*ny)/rr/pow((rr+dxrr[0]),2)*(1-pow(dxr[1],2)*(3*rr+dxrr[0])/pow(rr,2)/(rr+dxrr[0]))-
6*cc*xdash/pow(rr,5)*(1-5*pow(dxr[1],2)/pow(rr,2)));
sigma[2][2][1]=Ks*dxr[1]*((1-2*ny)*(3-4*ny)/pow(rr,3)-(9-12*ny)*pow(dxr[2],2)/pow(rr,5)-(4-4*ny)*(1-
2*ny)/rr/pow((rr+dxrr[0]),2)*(1-pow(dxr[2],2)*(3*rr+dxrr[0])/pow(rr,2)/(rr+dxrr[0]))+6*cc/pow(rr,5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/pow(rr,2)));
sigma[0][0][2]=dxr[2]*Ks*(-(1-2*ny)/pow(rr,3)-(9-12*ny)*pow(dxrr[0],2)/pow(rr,5)+6*cc/pow(rr,5)*(cc+(1-
2*ny)*dxrr[0]+5*xdash*pow(dxrr[0],2)/pow(rr,2)));
sigma[0][1][2]=Ks*dxr[1]*dxr[2]*(-(9-12*ny)*dxrr[0]/pow(rr,5)+6*cc/pow(rr,5)*(1-
2*ny+5*xdash*dxrr[0]/pow(rr,2)));
sigma[0][2][2]=Ks*((1-2*ny)*dxr[0]/pow(rr,3)-(9-12*ny)*pow(dxr[2],2)*dxrr[0]/pow(rr,5)-
6*cc/pow(rr,5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[2],2)-5*pow(dxr[2],2)*xdash*dxrr[0]/pow(rr,2)));
```

```
sigma[1][1][2]=Ks*dxr[2]*((1-2*ny)*(3-4*ny)/pow(rr,3)-(9-12*ny)*pow(dxr[1],2)/pow(rr,5)-(4-4*ny)*(1-
2*ny)/rr/pow((rr+dxrr[0]),2)*(1-pow(dxr[1],2)*(3*rr+dxrr[0])/pow(rr,2)/(rr+dxrr[0]))+6*cc/pow(rr,5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/pow(rr,2)));
sigma[1][2][2]=Ks*dxr[1]*((1-2*ny)/pow(rr,3)-(9-12*ny)*pow(dxr[2],2)/pow(rr,5)-(4-4*ny)*(1-
2*ny)/rr/pow((rr+dxrr[0]),2)*(1-pow(dxr[2],2)*(3*rr+dxrr[0])/pow(rr,2)/(rr+dxrr[0]))-
6*cc*xdash/pow(rr,5)*(1-5*pow(dxr[2],2)/pow(rr,2)));
sigma[2][2][2]=Ks*dxr[2]*((1-2*ny)*(5-4*ny)/pow(rr,3)-(9-12*ny)*pow(dxr[2],2)/pow(rr,5)-(4-4*ny)*(1-
2*ny)/rr/pow((rr+dxrr[0]),2)*(3-pow(dxr[2],2)*(3*rr+dxrr[0])/pow(rr,2)/(rr+dxrr[0]))+6*cc/pow(rr,5)*(3*cc-
(3-2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/pow(rr,2)));

sigma[1][0][0]=sigma[0][1][0];
sigma[2][0][0]=sigma[0][2][0];
sigma[2][1][0]=sigma[1][2][0];
sigma[1][0][1]=sigma[0][1][1];
sigma[2][0][1]=sigma[0][2][1];
sigma[2][1][1]=sigma[1][2][1];
sigma[1][0][2]=sigma[0][1][2];
sigma[2][0][2]=sigma[0][2][2];
sigma[2][1][2]=sigma[1][2][2];
for (int i = 0; i < Cd; ++i)
 for (int j = 0; j < Cd; ++j)
    {
        TMindlin[i][j]= 0;
    }
for (int i = 0; i < Cd; ++i)
 for (int j = 0; j < Cd; ++j)
  for (int k = 0; k < Cd; ++k)
    {
        TMindlin[i][j]= TMindlin[i][j]+sigma[j][k][i]*Vnor[k];
    }
return;
}

 void FishExample::SMelan(double **TSMelan,double *dxr,double *dxrr,double rr,double cc,double xdash,
                                                                    double  ny,double E,int
Cd)
 {
//-------------------------------------------------------------
//    MELAN SOLUTION FOR STRESS COMPUTATION
//    TO BE MULTIPLIED WITH T
//-------------------------------------------------------------
```

```
// Cd             :  Cartesian dimension (2D)
// dxr[size=cd]   :  rx,ry.
// dxrr[cd]       :  Rx,Ry.
// rr             :  R.
// cc             :  c.
// xdash          :  y¯.
// E              :  Young's modulus.
// ny             :  Poisson's ratio.
// TSMelan[2*cd][2*cd]: Melan solution derivatives to be multiplied with T.
double ***TSMelan1,***dUMelan; // Temporary arrays.
TSMelan1= new double**[Cd];
  for (int i = 0; i < Cd; ++i)
  {
    TSMelan1[i] = new double*[Cd];
    for (int j = 0; j < Cd; ++j)
      TSMelan1[i][j] = new double[Cd];
  }
dUMelan= new double**[Cd];
  for (int i = 0; i < Cd; ++i)
  {
    dUMelan[i] = new double*[Cd];
    for (int j = 0; j < Cd; ++j)
      dUMelan[i][j] = new double[Cd];
  }
double rr1=pow(rr,2);
double Pi=3.14159265359;
double G= E/(2.0*(1+ny));
double C= (2*G*ny)/(1-2*ny);
double Kd= 1.0/(8.0*Pi*G*(1.0 - ny));
int delta;
dUMelan[0][0][0] = Kd*((-8*pow((1-ny),2)+3-4*ny)/rr1*dxrr[0]+2*(3-4*ny)*dxrr[0]/rr1-2*((3-
4*ny)*pow(dxrr[0],2)-2*cc*xdash)/pow(rr1,2)*dxrr[0]+8*cc*xdash*dxrr[0]/pow(rr1,2)-
16*cc*xdash*pow(dxrr[0],3)/pow(rr1,3))+Kd*(-2*xdash/rr1+4*xdash*pow(dxrr[0],2)/pow(rr1,2));
dUMelan[0][1][0] =-Kd*(3-4*ny)*dxr[1]/rr1+Kd*(-2*(3-4*ny)*dxr[0]*dxr[1]/pow(rr1,2)*dxrr[0]
+4*cc*xdash*dxr[1]/pow(rr1,2)-16*cc*xdash*pow(dxrr[0],2)*dxr[1]/pow(rr1,3)+(4-4*ny)*(1-
2*ny)*dxrr[1]/pow(dxrr[0],2)/(1+pow(dxrr[1],2)/pow(dxrr[0],2)))+4*Kd*xdash*dxrr[0]*dxr[1]/pow(rr1,2);
dUMelan[1][0][0] =-Kd*(3-4*ny)*dxr[1]/rr1+Kd*(-2*(3-4*ny)*dxr[0]*dxr[1]/pow(rr1,2)*dxrr[0]-
4*cc*xdash*dxr[1]/pow(rr1,2)+16*cc*xdash*pow(dxrr[0],2)*dxr[1]/pow(rr1,3)-(4-4*ny)*(1-
2*ny)*dxrr[1]/pow(dxrr[0],2)/(1+pow(dxrr[1],2)/pow(dxrr[0],2)))-4*Kd*xdash*dxrr[0]*dxr[1]/pow(rr1,2);
dUMelan[1][1][0] =Kd*((-8*pow((1-ny),2)+3-4*ny)/rr1*dxrr[0]-2*((3-4*ny)*pow(dxr[1],2)
+2*cc*xdash)/pow(rr1,2)*dxrr[0]+16*cc*xdash*dxrr[0]*pow(dxr[1],2)/pow(rr1,3))+
```

```
Kd*(2*xdash/rr1-4*xdash*pow(dxr[1],2)/pow(rr1,2));
dUMelan[0][0][1] =-Kd*((-8*pow((1-ny),2)+3-4*ny)/rr1*dxrr[1]-2*((3-4*ny)*pow(dxrr[0],2)-
2*cc*xdash)/pow(rr1,2)*dxrr[1]-16*cc*xdash*pow(dxrr[0],2)/pow(rr1,3)*dxrr[1]);
dUMelan[0][1][1] =-Kd*((3-4*ny)*dxr[0]/rr1+4*cc*xdash*dxrr[0]/pow(rr1,2))-Kd*(-2*(3-
4*ny)*dxr[0]*dxr[1]/pow(rr1,2)*dxrr[1]-16*cc*xdash*dxrr[0]*dxr[1]/pow(rr1,3)*dxrr[1]-(4-4*ny)*(1-
2*ny)/dxrr[0]/(1+pow(dxrr[1],2)/pow(dxrr[0],2)));
dUMelan[1][0][1] =-Kd*((3-4*ny)*dxr[0]/rr1-4*cc*xdash*dxrr[0]/pow(rr1,2))-Kd*(-2*(3-
4*ny)*dxr[0]*dxr[1]/pow(rr1,2)*dxrr[1]+16*cc*xdash*dxrr[0]*dxr[1]/pow(rr1,3)*dxrr[1]+(4-4*ny)*(1-
2*ny)/dxrr[0]/(1+pow(dxrr[1],2)/pow(dxrr[0],2)));
dUMelan[1][1][1] =-Kd*(2*(3-4*ny)*dxr[1]/rr1-8*cc*xdash*dxr[1]/pow(rr1,2))-Kd*((-8*pow((1-ny),2)+3-
4*ny)/rr1*dxrr[1]-2*((3-4*ny)*pow(dxr[1],2)+2*cc*xdash)/pow(rr1,2)*dxrr[1]
+16*cc*xdash*pow(dxr[1],2)/pow(rr1,3)*dxrr[1]);
for (int i = 0; i < Cd; ++i)
 for (int j = 0; j < Cd; ++j)
  for (int k = 0; k < Cd; ++k)
    {
        if (i==j)
         {
            delta=1;
         }
        else
        {
           delta=0;
        }
       TSMelan1[i][j][k]=G*(dUMelan[i][k][j]+dUMelan[j][k][i])+C*delta*(dUMelan[0][k][0]+dUMelan[1][k][1]);
    }
TSMelan[0][0] =TSMelan1[0][0][0];
TSMelan[0][1] =TSMelan1[0][0][1];
TSMelan[1][0] =TSMelan1[1][1][0];
TSMelan[1][1] =TSMelan1[1][1][1];
TSMelan[2][0] =TSMelan1[0][1][0];
TSMelan[2][1] =TSMelan1[0][1][1];
TSMelan[3][0] =0;
TSMelan[3][1] =0;

return;
}
```

```
void FishExample::RMelan(double **USMelan,double *dxr,double *Vnor,double *dxrr,double rr,double cc,
                                                                    double xdash,double ny,double E,int
Cd)
{
//-------------------------------------------------------------
//    MELAN SOLUTION FOR STRESS COMPUTATION
//    TO BE MULTIPLIED WITH U
//-------------------------------------------------------------
// Cd                :  Cartesian dimension (2D)
// dxr[size=cd]      :  rx,ry.
// dxrr[cd]          :  Rx,Ry.
// Vnor[cd]          :  normal vector.
// rr                :  R.
// cc                :  c.
// xdash             :  y¯.
// E                 :  Young's modulus.
// ny                :  Poisson's ratio.
// USMelan[2*cd][2*cd]: Melan solution derivatives to be multiplied with U.
double ***USMelan1,****dSigma; // Temporary arrays.
USMelan1= new double**[Cd];
  for (int i = 0; i < Cd; ++i)
  {
    USMelan1[i] = new double*[Cd];
    for (int j = 0; j < Cd; ++j)
      USMelan1[i][j] = new double[Cd];
  }
dSigma= new double***[Cd];
  for (int i = 0; i < Cd; ++i)
  {
   dSigma[i] = new double**[Cd];
    for (int j = 0; j < Cd; ++j)
      {
      dSigma[i][j] = new double*[Cd];
      for (int k = 0; k < Cd; ++k)
         {
       dSigma[i][j][k] = new double[Cd];
       }
    }
   }
int delta;
double rr1=pow(rr,2);
```

```
double Pi=3.14159265359;
double Ks=1/(4*Pi*(1-ny));
double C=(2*ny)/(1-2*ny);
double G= E/(2.0*(1+ny));
  for (int i = 0; i < Cd; ++i)
    for (int j = 0; j < Cd; ++j)
     for (int k = 0; k < Cd; ++k)
  {
       USMelan1[i][j][k] = 0;
  }
dSigma[0][0][0][0]=-Ks*(-2*(3*xdash+cc)*(1-
2*ny)/pow(rr1,2)*dxrr[0]+(6*pow(dxrr[0],2)+4*cc*xdash)/pow(rr1,2)-4*(2*dxrr[0]*(pow(dxrr[0],2)+2*cc*xdash)-
4*xdash*pow(dxr[1],2)*(1-2*ny))/pow(rr1,3)*dxrr[0]-
16*cc*xdash*pow(dxr[1],2)/pow(rr1,3)+96*cc*xdash*pow(dxrr[0],2)*pow(dxr[1],2)/pow(rr1,4))
-Ks*((1-2*ny)/rr1+4*xdash*dxrr[0]/pow(rr1,2)-16*xdash*dxrr[0]*pow(dxr[1],2)/pow(rr1,3));

dSigma[0][1][0][0]=-Ks*dxr[1]*(-2*(-1+2*ny)/pow(rr1,2)*dxrr[0]+4*xdash*(1-2*ny)/pow(rr1,2)-
4*(2*pow(xdash,2)-4*cc*xdash-2*pow(cc,2)+4*xdash*dxrr[0]*(1-
2*ny))/pow(rr1,3)*dxrr[0]+32*cc*xdash*dxrr[0]/pow(rr1,3)-96*cc*xdash*pow(dxrr[0],3)/pow(rr1,4))-
Ks*dxr[1]*((-4*xdash-4*cc)/pow(rr1,2)+16*xdash*pow(dxrr[0],2)/pow(rr1,3));

dSigma[1][1][0][0]=-Ks*(-2*(xdash+3*cc)*(1-
2*ny)/pow(rr1,2)*dxrr[0]+(2*pow(dxr[1],2)+4*pow(cc,2))/pow(rr1,2)-4*(2*dxrr[0]*(pow(dxr[1],2)+2*pow(cc,2))-
4*cc*pow(dxr[1],2)+4*xdash*pow(dxr[1],2)*(1-2*ny))/pow(rr1,3)*dxrr[0]+16*cc*xdash*pow(dxr[1],2)/pow(rr1,3)-
96*cc*xdash*pow(dxrr[0],2)*pow(dxr[1],2)/pow(rr1,4))-Ks*(3*(1-2*ny)/rr1+(8*dxrr[0]*cc-
4*pow(dxr[1],2))/pow(rr1,2)+16*xdash*dxrr[0]*pow(dxr[1],2)/pow(rr1,3));

dSigma[0][0][1][0]=-Ks*dxr[1]*(-2*(1-2*ny)/pow(rr1,2)*dxrr[0]+4*xdash*(1-2*ny)/pow(rr1,2)+4*(2*pow(cc,2)-
2*pow(xdash,2)+12*cc*xdash-4*xdash*dxrr[0]*(1-2*ny))
/pow(rr1,3)*dxrr[0]-96*cc*xdash*pow(dxr[1],2)/pow(rr1,4)*dxrr[0])-Ks*dxr[1]*(-
(4*cc+12*xdash)/pow(rr1,2)+16*xdash*pow(dxr[1],2)/pow(rr1,3));

dSigma[0][1][1][0]=-Ks*(-2*(3*xdash+cc)*(1-2*ny)/pow(rr1,2)*dxrr[0]+(4*cc*xdash+2*pow(dxr[1],2)-
8*xdash*dxrr[0]*(1-2*ny))/pow(rr1,2)-4*(2*(2*cc*xdash+pow(dxr[1],2))*dxrr[0]
-4*xdash*pow(dxrr[0],2)*(1-2*ny))/pow(rr1,3)*dxrr[0]-
16*cc*xdash*pow(dxr[1],2)/pow(rr1,3)+96*cc*xdash*pow(dxrr[0],2)*pow(dxr[1],2)/pow(rr1,4))-Ks*((1-2*ny)/rr1
+4*xdash*dxrr[0]/pow(rr1,2)-16*xdash*dxrr[0]*pow(dxr[1],2)/pow(rr1,3));

dSigma[1][1][1][0]=-Ks*dxr[1]*(-2*(3-6*ny)/pow(rr1,2)*dxrr[0]-4*xdash*(1-2*ny)/pow(rr1,2)-
4*(2*pow(dxr[1],2)-8*cc*xdash-4*pow(cc,2)-4*xdash*dxrr[0]*(1-2*ny))
```

```
/pow(rr1,3)*dxrr[0]+32*cc*xdash*dxrr[0]/pow(rr1,3)-96*cc*xdash*pow(dxrr[0],3)/pow(rr1,4))-Ks*dxr[1]*((-
8*xdash-8*cc)/pow(rr1,2)+16*xdash*pow(dxrr[0],2)/pow(rr1,3));

dSigma[0][0][0][1]=Ks*(-8*xdash*dxr[1]*(1-2*ny)/pow(rr1,2)-32*cc*xdash*dxrr[0]*dxr[1]/pow(rr1,3))+Ks*(-
2*(3*xdash+cc)*(1-2*ny)/pow(rr1,2)*dxrr[1]-4*(2*dxrr[0]*(pow(dxrr[0],2)+2*cc*xdash)-
4*xdash*pow(dxr[1],2)*(1-2*ny))/pow(rr1,3)*dxrr[1]+96*cc*xdash*dxrr[0]*pow(dxr[1],2)/pow(rr1,4)*dxrr[1]);

dSigma[0][1][0][1]=Ks*((-1+2*ny)/rr1+(2*pow(xdash,2)-4*cc*xdash-2*pow(cc,2)+4*xdash*dxrr[0]*(1-
2*ny))/pow(rr1,2)+16*cc*xdash*pow(dxrr[0],2)/pow(rr1,3))
+Ks*dxr[1]*(-2*(-1+2*ny)/pow(rr1,2)*dxrr[1]-4*(2*pow(xdash,2)-4*cc*xdash-2*pow(cc,2)+4*xdash*dxrr[0]*(1-
2*ny))/pow(rr1,3)*dxrr[1]-96*cc*xdash*pow(dxrr[0],2)/pow(rr1,4)*dxrr[1]);

dSigma[1][1][0][1] =Ks*((4*dxrr[0]*dxr[1]-8*cc*dxr[1]+8*xdash*dxr[1]*(1-
2*ny))/pow(rr1,2)+32*cc*xdash*dxrr[0]*dxr[1]/pow(rr1,3))+Ks*(-2*(xdash+3*cc)*(1-2*ny)
/pow(rr1,2)*dxrr[1]-4*(2*dxrr[0]*(pow(dxr[1],2)+2*pow(cc,2))-4*cc*pow(dxr[1],2)+4*xdash*pow(dxr[1],2)*(1-
2*ny))/pow(rr1,3)*dxrr[1]-96*cc*xdash*dxrr[0]*pow(dxr[1],2)/pow(rr1,4)*dxrr[1]);

dSigma[0][0][1][1]=Ks*((1-2*ny)/rr1-(2*pow(cc,2)-2*pow(xdash,2)+12*cc*xdash-4*xdash*dxrr[0]*(1-
2*ny))/pow(rr1,2)+16*cc*xdash*pow(dxr[1],2)/pow(rr1,3))+32*Ks*pow(dxr[1],2)*cc*xdash
/pow(rr1,3)+Ks*dxr[1]*(-2*(1-2*ny)/pow(rr1,2)*dxrr[1]+4*(2*pow(cc,2)-2*pow(xdash,2)+12*cc*xdash-
4*xdash*dxrr[0]*(1-2*ny))/pow(rr1,3)*dxrr[1]-96*cc*xdash*pow(dxr[1],2)/pow(rr1,4)*dxrr[1]);

dSigma[0][1][1][1]=Ks*(4*dxrr[0]*dxr[1]/pow(rr1,2)-32*cc*xdash*dxrr[0]*dxr[1]/pow(rr1,3))+Ks*(-
2*(3*xdash+cc)*(1-2*ny)/pow(rr1,2)*dxrr[1]-4*(2*(2*cc*xdash+pow(dxr[1],2))*dxrr[0]
-4*xdash*pow(dxrr[0],2)*(1-2*ny))/pow(rr1,3)*dxrr[1]+96*cc*xdash*dxrr[0]*pow(dxr[1],2)/pow(rr1,4)*dxrr[1]);

dSigma[1][1][1][1] =Ks*((3-6*ny)/rr1+(2*pow(dxr[1],2)-8*cc*xdash-4*pow(cc,2)-4*xdash*dxrr[0]*(1-
2*ny))/pow(rr1,2)+16*cc*xdash*pow(dxrr[0],2)/pow(rr1,3))+4*Ks*pow(dxr[1],2)/pow(rr1,2)
+Ks*dxr[1]*(-2*(3-6*ny)/pow(rr1,2)*dxrr[1]-4*(2*pow(dxr[1],2)-8*cc*xdash-4*pow(cc,2)-4*xdash*dxrr[0]*(1-
2*ny))/pow(rr1,3)*dxrr[1]-96*cc*xdash*pow(dxrr[0],2)/pow(rr1,4)*dxrr[1]);

dSigma[1][0][0][0]=dSigma[0][1][0][0];
dSigma[1][0][1][0]=dSigma[0][1][1][0];
dSigma[1][0][0][1]=dSigma[0][1][0][1];
dSigma[1][0][1][1]=dSigma[0][1][1][1];
for (int i = 0; i < Cdim; ++i)
 for (int j = 0; j < Cdim; ++j)
  for (int k = 0; k < Cdim; ++k)
   for (int m = 0; m < Cdim; ++m)
    {
        if (i==j)
```

```
          {
              delta=1;
          }
         else
         {
            delta=0;
         }

USMelan1[i][j][k]=USMelan1[i][j][k]+G*(dSigma[k][m][i][j]+dSigma[k][m][j][i]+C*delta*(dSigma[k][m][0][0]+dS
igma[k][m][1][1]))*Vnor[m];
     }
USMelan[0][0] =USMelan1[0][0][0];
USMelan[0][1] =USMelan1[0][0][1];
USMelan[1][0] =USMelan1[1][1][0];
USMelan[1][1] =USMelan1[1][1][1];
USMelan[2][0] =USMelan1[0][1][0];
USMelan[2][1] =USMelan1[0][1][1];
USMelan[3][0] =0;
USMelan[3][1] =0;
return;
}

void FishExample::SMindlin(double **TSMindlin,double *dxr,double *dxrr,double rr,double cc,double xdash,
                                                                          double  ny,double E,int
Cdim)
{
//-------------------------------------------------------------
//    MINDLIN SOLUTION FOR STRESS COMPUTATION
//    TO BE MULTIPLIED WITH T
//-------------------------------------------------------------
// Cd              :  Cartesian dimension (3D )
// dxr[size=cd]    :  rx,ry,rz.
// dxrr[cd]        :  Rx,Ry,Rz.
// rr              :  R.
// cc              :  c.
// xdash           :  y-
// E               :  Young's modulus.
// ny              :  Poisson's ratio.
// TSMindlin[size= 2*Cd][Cd]:  Mindlin solution derivatives to be multiplied with T.
double ***dUMindlin, ***TSMindlin1,delta; // Temporary arrays.
dUMindlin= new double**[Cd];
```

```
  for (int i = 0; i < Cd; ++i)
  {
    dUMindlin[i] = new double*[Cd];
    for (int j = 0; j < Cd; ++j)
      dUMindlin[i][j] = new double[Cd];
  }
TSMindlin1= new double**[Cd];
  for (int i = 0; i < Cd; ++i)
  {
    TSMindlin1[i] = new double*[Cd];
    for (int j = 0; j < Cd; ++j)
      TSMindlin1[i][j] = new double[Cd];
  }
double rr1=pow(rr,2);
double Pi=3.14159265359;
double G= E/(2.0*(1+ny));
double C= (2*G*ny)/(1-2*ny);
double Kd= 1.0/(16.0*Pi*G*(1.0 - ny));
dUMindlin[0][0][0] =Kd*(-(8*pow((1-ny),2)-3+4*ny)/pow(rr1,1.5)*dxrr[0]+2*(3-4*ny)*dxrr[0]/pow(rr1,1.5)-
3*((3-4*ny)*pow(dxrr[0],2)-2*cc*xdash)/pow(rr1,2.5)*dxrr[0]+12*cc*xdash*dxrr[0]/pow(rr1,2.5)-
30*cc*xdash*pow(dxrr[0],3)/pow(rr1,3.5))+Kd*(-2*xdash/pow(rr1,1.5)+6*xdash*pow(dxrr[0],2)/pow(rr1,2.5));

dUMindlin[0][1][0] =-Kd*dxr[1]*(3-4*ny)/pow(rr1,1.5)+Kd*dxr[1]*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[0]+(4-
4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)+6*cc*xdash/pow(rr1,2.5)-
30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5))+6*Kd*dxr[1]*xdash*dxrr[0]/pow(rr1,2.5);

dUMindlin[0][2][0] =-dxr[2]*Kd*(3-4*ny)/pow(rr1,1.5)+dxr[2]*Kd*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[0]+(4-
4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)+6*cc*xdash/pow(rr1,2.5)-
30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5))+6*dxr[2]*Kd*xdash*dxrr[0]/pow(rr1,2.5);

dUMindlin[1][0][0] =-Kd*dxr[1]*(3-4*ny)/pow(rr1,1.5)+Kd*dxr[1]*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[0]-(4-
4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]
-(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)-
6*cc*xdash/pow(rr1,2.5)+30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5))-6*Kd*dxr[1]*xdash*dxrr[0]/pow(rr1,2.5);

dUMindlin[1][1][0] =Kd*(-1/pow(rr1,1.5)*dxrr[0]-3*(3-4*ny)*pow(dxr[1],2)/pow(rr1,2.5)*dxrr[0]-
6*cc*xdash/pow(rr1,2.5)*(1-3*pow(dxr[1],2)/rr1)*dxrr[0]+12*cc*xdash/pow(rr1,3.5)*pow(dxr[1],2)*dxrr[0]-(4-
4*ny)*(1-2*ny)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)+(4-4*ny)*(1-
```

```
2*ny)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[1],2)/pow(r
r1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)))+2*Kd*xdash/pow(rr1,1.5)*(1-
3*pow(dxr[1],2)/rr1);

dUMindlin[1][2][0] =Kd*dxr[1]*dxr[2]*(-3*(3-4*ny)/pow(rr1,2.5)*dxrr[0]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[0]+2*(4-4*ny)*(1-2*ny)
/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*(1/pow(rr1,0.5)*dxrr[0]+1)+30*cc*xdash/pow(rr1,3.5)*dxrr[0])-
6*Kd*dxr[1]*dxr[2]*xdash/pow(rr1,2.5);

dUMindlin[2][0][0] =-dxr[2]*Kd*(3-4*ny)/pow(rr1,1.5)+dxr[2]*Kd*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[0]-(4-
4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]
-(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)-
6*cc*xdash/pow(rr1,2.5)+30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5))-6*dxr[2]*Kd*xdash*dxrr[0]/pow(rr1,2.5);

dUMindlin[2][1][0] =Kd*dxr[1]*dxr[2]*(-3*(3-4*ny)/pow(rr1,2.5)*dxrr[0]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[0]+2*(4-4*ny)*(1-2*ny)
/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*(1/pow(rr1,0.5)*dxrr[0]+1)+30*cc*xdash/pow(rr1,3.5)*dxrr[0])-
6*Kd*dxr[1]*dxr[2]*xdash/pow(rr1,2.5);

dUMindlin[2][2][0] =Kd*(-1/pow(rr1,1.5)*dxrr[0]-3*(3-4*ny)*pow(dxr[2],2)/pow(rr1,2.5)*dxrr[0]-
6*cc*xdash/pow(rr1,2.5)*(1-3*pow(dxr[2],2)/rr1)*dxrr[0]+12*cc*xdash
/pow(rr1,3.5)*pow(dxr[2],2)*dxrr[0]-(4-4*ny)*(1-2*ny)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)+(4-4*ny)*(1-
2*ny)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[2],2)/pow(r
r1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)))+2*Kd*xdash/pow(rr1,1.5)*(1-
3*pow(dxr[2],2)/rr1);

dUMindlin[0][0][1] =-Kd*(-(8*pow((1-ny),2)-3+4*ny)/pow(rr1,1.5)*dxrr[1]-3*((3-4*ny)*pow(dxrr[0],2)-
2*cc*xdash)/pow(rr1,2.5)*dxrr[1]-30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5)*dxrr[1]);

dUMindlin[0][1][1] =-Kd*((3-4*ny)*dxr[0]/pow(rr1,1.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])+6*cc*xdash*dxrr[0]/pow(rr1,2.5))-Kd*dxr[1]*(-3*(3-4*ny)*dxr[0]
/pow(rr1,2.5)*dxrr[1]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]-30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[1]);

dUMindlin[0][2][1] =-dxr[2]*Kd*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[1]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]
-30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[1]);
```

```
dUMindlin[1][0][1] =-Kd*((3-4*ny)*dxr[0]/pow(rr1,1.5)+(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-6*cc*xdash*dxrr[0]/pow(rr1,2.5))-Kd*dxr[1]*(-3*(3-4*ny)*dxr[0]
/pow(rr1,2.5)*dxrr[1]-(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]-(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]+30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[1]);

dUMindlin[1][1][1] =-Kd*(2*(3-4*ny)*dxr[1]/pow(rr1,1.5)-12*cc*xdash/pow(rr1,2.5)*dxr[1]-2*(4-4*ny)*(1-
2*ny)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxr[1]/pow(rr1,0.5))-Kd*(-1/pow(rr1,1.5)*dxrr[1]
-3*(3-4*ny)*pow(dxr[1],2)/pow(rr1,2.5)*dxrr[1]-6*cc*xdash/pow(rr1,2.5)*(1-
3*pow(dxr[1],2)/rr1)*dxrr[1]+12*cc*xdash/pow(rr1,3.5)*pow(dxr[1],2)*dxrr[1]-(4-4*ny)*(1-2*ny)
/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0]))/pow(rr1,0.5)*dxrr[1]+(4-4*ny)*(1-
2*ny)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[1],2)/pow(rr1,1.5)
/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dxr[1],2)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]));

dUMindlin[1][2][1] =-Kd*dxr[2]*((3-4*ny)/pow(rr1,1.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)-6*cc*xdash/pow(rr1,2.5))-Kd*dxr[1]*dxr[2]*(-3*(3-
4*ny)/pow(rr1,2.5)*dxrr[1]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]+2*(4-
4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*dxrr[1]+30*cc*xdash/pow(rr1,3.5)*dxrr[1]);

dUMindlin[2][0][1] =-dxr[2]*Kd*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[1]-(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]-(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]
+30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[1]);

dUMindlin[2][1][1] =-Kd*dxr[2]*((3-4*ny)/pow(rr1,1.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)-6*cc*xdash/pow(rr1,2.5))-Kd*dxr[1]*dxr[2]*(-3*(3-
4*ny)/pow(rr1,2.5)*dxrr[1]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]+2*(4-
4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*dxrr[1]+30*cc*xdash/pow(rr1,3.5)*dxrr[1]);

dUMindlin[2][2][1] =-Kd*(-1/pow(rr1,1.5)*dxrr[1]-3*(3-4*ny)*pow(dxr[2],2)/pow(rr1,2.5)*dxrr[1]-
6*cc*xdash/pow(rr1,2.5)*(1-3*pow(dxr[2],2)/rr1)*dxrr[1]+12*cc*xdash/pow(rr1,3.5)*pow(dxr[2],2)*dxrr[1]
-(4-4*ny)*(1-2*ny)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0]))/pow(rr1,0.5)*dxrr[1]+(4-4*ny)*(1-
2*ny)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dxr[2],2)/rr1/p
ow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]));

dUMindlin[0][0][2] =-Kd*(-(8*pow((1-ny),2)-3+4*ny)/pow(rr1,1.5)*dxrr[2]-3*((3-4*ny)*pow(dxrr[0],2)-
2*cc*xdash)/pow(rr1,2.5)*dxrr[2]-30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5)*dxrr[2]);
```

```
dUMindlin[0][1][2] =-Kd*dxr[1]*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[2]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]-30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[2]);

dUMindlin[0][2][2] =-Kd*((3-4*ny)*dxr[0]/pow(rr1,1.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])+6*cc*xdash*dxrr[0]/pow(rr1,2.5))-dxr[2]*Kd*(-3*(3-4*ny)*dxr[0]
/pow(rr1,2.5)*dxrr[2]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]-30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[2]);

dUMindlin[1][0][2] =-Kd*dxr[1]*(-3*(3-4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]-(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]
+30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[2]);

dUMindlin[1][1][2] =-Kd*(-1/pow(rr1,1.5)*dxrr[2]-3*(3-4*ny)*pow(dxr[1],2)/pow(rr1,2.5)*dxrr[2]-
6*cc*xdash/pow(rr1,2.5)*(1-3*pow(dxr[1],2)/rr1)*dxrr[2]+12*cc*xdash/pow(rr1,3.5)*pow(dxr[1],2)*dxrr[2]
-(4-4*ny)*(1-2*ny)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0]))/pow(rr1,0.5)*dxrr[2]+(4-4*ny)*(1-
2*ny)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+pow(dxr[1],2)/rr1/p
ow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]));

dUMindlin[1][2][2] =-Kd*dxr[1]*((3-4*ny)/pow(rr1,1.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)-6*cc*xdash/pow(rr1,2.5))-Kd*dxr[1]*dxr[2]*(-3*(3-
4*ny)/pow(rr1,2.5)*dxrr[2]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]+2*(4-
4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*dxrr[2]+30*cc*xdash/pow(rr1,3.5)*dxrr[2]);

dUMindlin[2][0][2] =-Kd*((3-4*ny)*dxr[0]/pow(rr1,1.5)+(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-6*cc*xdash*dxrr[0]/pow(rr1,2.5))-dxr[2]*Kd*(-3*(3-
4*ny)*dxr[0]/pow(rr1,2.5)*dxrr[2]-(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]-(4-
4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]+30*cc*xdash*dxrr[0]/pow(rr1,3.5)*dxrr[2]);

dUMindlin[2][1][2] =-Kd*dxr[1]*((3-4*ny)/pow(rr1,1.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)-6*cc*xdash/pow(rr1,2.5))-Kd*dxr[1]*dxr[2]*(-3*(3-
4*ny)/pow(rr1,2.5)*dxrr[2]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]+2*(4-
4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*dxrr[2]+30*cc*xdash/pow(rr1,3.5)*dxrr[2]);

dUMindlin[2][2][2] =-Kd*(2*(3-4*ny)*dxr[2]/pow(rr1,1.5)-12*cc*xdash/pow(rr1,2.5)*dxr[2]-2*(4-4*ny)*(1-
2*ny)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxr[2]/pow(rr1,0.5))-Kd*(-1/pow(rr1,1.5)*dxrr[2]-3*(3-
4*ny)*pow(dxr[2],2)/pow(rr1,2.5)*dxrr[2]-6*cc*xdash/pow(rr1,2.5)*(1-
3*pow(dxr[2],2)/rr1)*dxrr[2]+12*cc*xdash/pow(rr1,3.5)*pow(dxr[2],2)*dxrr[2]-(4-4*ny)*(1-2*ny)
```

```
/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0]))/pow(rr1,0.5)*dxrr[2]+(4-4*ny)*(1-
2*ny)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[2],2)
/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+pow(dxr[2],2)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]));

for (int i = 0; i < Cd; ++i)
 for (int j = 0; j < Cd; ++j)
  for (int k = 0; k < Cd; ++k)
    {
       TSMindlin1[i][j][k]=0.0;
     }
for (int i = 0; i < Cd; ++i)
 for (int j = 0; j < Cd; ++j)
  for (int k = 0; k < Cd; ++k)
    {
        if (i==j)
          {
             delta=1;
          }
        else
          {
           delta=0;
          }

TSMindlin1[i][j][k]=G*(dUMindlin[i][k][j]+dUMindlin[j][k][i])+C*delta*(dUMindlin[0][k][0]+dUMindlin[1][k][1
]+dUMindlin[2][k][2]);
     }
TSMindlin[0][0] =TSMindlin1[0][0][0];
TSMindlin[0][1] =TSMindlin1[0][0][1];
TSMindlin[0][2] =TSMindlin1[0][0][2];
TSMindlin[1][0] =TSMindlin1[1][1][0];
TSMindlin[1][1] =TSMindlin1[1][1][1];
TSMindlin[1][2] =TSMindlin1[1][1][2];
TSMindlin[2][0] =TSMindlin1[2][2][0];
TSMindlin[2][1] =TSMindlin1[2][2][1];
TSMindlin[2][2] =TSMindlin1[2][2][2];
TSMindlin[3][0] =TSMindlin1[0][1][0];
TSMindlin[3][1] =TSMindlin1[0][1][1];
TSMindlin[3][2] =TSMindlin1[0][1][2];
TSMindlin[4][0] =TSMindlin1[1][2][0];
TSMindlin[4][1] =TSMindlin1[1][2][1];
```

```
TSMindlin[4][2] =TSMindlin1[1][2][2];
TSMindlin[5][0] =TSMindlin1[0][2][0];
TSMindlin[5][1] =TSMindlin1[0][2][1];
TSMindlin[5][2] =TSMindlin1[0][2][2];
return;
}
void FishExample::RMindlin(double **USMindlin,double *dxr,double *Vnor,double *dxrr,double rr,double cc,
                                                                    double xdash,double ny,double
E,int Cd)
{
//-------------------------------------------------------------
//    MINDLIN SOLUTION FOR STRESS COMPUTATION
//    TO BE MULTIPLIED WITH U
//-------------------------------------------------------------
// Cd              :  Cartesian dimension (3D )
// dxr[size=cd]    :  rx,ry,rz.
// dxrr[cd]        :  Rx,Ry,Rz.
// Vnor[cd]        :  normal vector.
// rr              :  R.
// cc              :  c.
// xdash           :  y-
// E               :  Young's modulus.
// ny              :  Poisson's ratio.
// USMindlin[2*Cd][Cd]:  Mindlin solution derivatives to be multiplied with U.
double rr1=pow(rr,2);
double Pi=3.14159265359;
double Ks=1/(8*Pi*(1-ny));
double C=(2*ny)/(1-2*ny);
double G= E/(2.0*(1+ny));
double ***USMindlin1,****dsigma,delta; // Temporary arrays
USMindlin1= new double**[Cdim];
  for (int i = 0; i < Cdim; ++i)
  {
    USMindlin1[i] = new double*[Cdim];
    for (int j = 0; j < Cdim; ++j)
      USMindlin1[i][j] = new double[Cdim];
  }
dsigma= new double***[Cdim];
  for (int i = 0; i < Cdim; ++i)
  {
   dsigma[i] = new double**[Cdim];
```

```
    for (int j = 0; j < Cdim; ++j)
      {
      dsigma[i][j] = new double*[Cdim];
      for (int k = 0; k < Cdim; ++k)
         {
       dsigma[i][j][k] = new double[Cdim];
       }
    }
   }
dsigma[0][0][0][0] =-Ks*(1-2*ny)/pow(rr1,1.5)+Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[0]-(2*(9-
12*ny)*xdash*dxrr[0]-3*cc*(5*xdash-cc))/pow(rr1,2.5)+5*((9-12*ny)*xdash*pow(dxrr[0],2)-
3*cc*dxrr[0]*(5*xdash-cc))/pow(rr1,3.5)*dxrr[0]-
90*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5)+210*cc*xdash*pow(dxrr[0],4)
/pow(rr1,4.5))+Ks*(-(-3*dxrr[0]*(5*xdash-cc)+3*cc*dxrr[0])/pow(rr1,2.5)-
30*xdash*pow(dxrr[0],3)/pow(rr1,3.5));

dsigma[0][1][0][0] =Ks*dxr[1]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[0]-(9-12*ny)*xdash/pow(rr1,2.5)+5*((9-
12*ny)*xdash*dxrr[0]-3*cc*(3*xdash+cc))/pow(rr1,3.5)*dxrr[0]
-60*cc*xdash*dxrr[0]/pow(rr1,3.5)+210*cc*xdash*pow(dxrr[0],3)/pow(rr1,4.5))+Ks*dxr[1]*(-(-9*xdash-
6*cc)/pow(rr1,2.5)-30*xdash*pow(dxrr[0],2)/pow(rr1,3.5));

dsigma[0][2][0][0] =Ks*dxr[2]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[0]-(9-12*ny)*xdash/pow(rr1,2.5)+5*((9-
12*ny)*xdash*dxrr[0]-3*cc*(3*xdash+cc))/pow(rr1,3.5)*dxrr[0]
-60*cc*xdash*dxrr[0]/pow(rr1,3.5)+210*cc*xdash*pow(dxrr[0],3)/pow(rr1,4.5))+Ks*dxr[2]*(-(-9*xdash-
6*cc)/pow(rr1,2.5)-30*xdash*pow(dxrr[0],2)/pow(rr1,3.5));

dsigma[1][1][0][0] =-Ks*(3*(1-2*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[1],2)/pow(rr1,2.5))+Ks*(-4*(1-
2*ny)*ny/pow(rr1,1.5)-3*(1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr1,2.5)*dxrr[0]+6*cc*((1-2*ny)*xdash-
2*ny*cc)/pow(rr1,2.5)+5*((9-12*ny)*pow(dxr[1],2)*dxr[0]-6*cc*dxrr[0]*((1-2*ny)*xdash-2*ny*cc))
/pow(rr1,3.5)*dxrr[0]-30*cc*pow(dxr[1],2)*xdash/pow(rr1,3.5)
+210*cc*pow(dxr[1],2)*xdash*pow(dxrr[0],2)/pow(rr1,4.5)+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)
/(pow(rr1,0.5)+dxrr[0])*(1-pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-pow(dxr[1],2)/rr1)*dxrr[0]+(4-
4*ny)*(1-2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-pow(dxr[1],2)/rr1)*(1/pow(rr1,0.5)*dxrr[0]+1)-(4-
4*ny)*(1-2*ny)
/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[1],
2)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)+2*pow(dxr[1],2)/pow(rr1,2)*dxrr[0]
))+Ks*(-(-6*dxrr[0]*((1-2*ny)*xdash-2*ny*cc)+12*cc*dxrr[0]*ny)/pow(rr1,2.5)-
30*pow(dxr[1],2)*xdash*dxrr[0]/pow(rr1,3.5));
```

```
dsigma[1][2][0][0] =-Ks*dxr[1]*dxr[2]*(-9+12*ny)/pow(rr1,2.5)+Ks*dxr[1]*dxr[2]*(-5*(-
9+12*ny)*dxr[0]/pow(rr1,3.5)*dxrr[0]-2*(4-4*ny)*(1-2*ny)/pow(rr1,2)
/(pow(rr1,0.5)+dxrr[0])*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))*dxrr[0]-(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))*(1/pow(rr1,0.5)*dxrr[0]+1
)+(4-4*ny)*(1-2*ny)/rr1/(pow(rr1,0.5)+dxrr[0])*(-1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)
-1/pow(rr1,1.5)*dxrr[0])-30*cc*xdash/pow(rr1,3.5)+210*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5))-
30*Ks*dxr[1]*dxr[2]*xdash*dxrr[0]/pow(rr1,3.5);

dsigma[2][2][0][0] =-Ks*(3*(1-2*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[2],2)/pow(rr1,2.5))+Ks*(-4*(1-
2*ny)*ny/pow(rr1,1.5)-3*(1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr1,2.5)*dxrr[0]+6*cc*((1-2*ny)*xdash-
2*ny*cc)/pow(rr1,2.5)+5*((9-12*ny)*pow(dxr[2],2)*dxr[0]-6*cc*dxrr[0]*((1-2*ny)*xdash-2*ny*cc))
/pow(rr1,3.5)*dxrr[0]-30*cc*pow(dxr[2],2)*xdash/pow(rr1,3.5)+
210*cc*pow(dxr[2],2)*xdash*pow(dxrr[0],2)/pow(rr1,4.5)+(4-4*ny)*(1-2*ny)
/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*(1-pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-
pow(dxr[2],2)/rr1)*dxrr[0]+(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-pow(dxr[2],2)/rr1)*(1/pow(rr1,0.5)*dxrr[0]+1)-(4-
4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dx
r[2],2)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1)+2*pow(dxr[2],2)/pow(rr1,2)*dx
rr[0]))+Ks*(-(-6*dxrr[0]*((1-2*ny)*xdash-2*ny*cc)
+12*cc*dxrr[0]*ny)/pow(rr1,2.5)-30*pow(dxr[2],2)*xdash*dxrr[0]/pow(rr1,3.5));

dsigma[0][0][1][0] =Ks*dxr[1]*(-3*(-1+2*ny)/pow(rr1,2.5)*dxrr[0]-2*(9-12*ny)*dxrr[0]/pow(rr1,2.5)+5*(9-
12*ny)*pow(dxrr[0],3)/pow(rr1,3.5)-30*cc/pow(rr1,3.5)
*(cc+(1-2*ny)*dxrr[0]+5*xdash*pow(dxrr[0],2)/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(1-2*ny+10*xdash*dxrr[0]/rr1-
10*xdash*pow(dxrr[0],3)/pow(rr1,2)))+Ks*dxr[1]*(6/pow(rr1,2.5)*(cc+(1-
2*ny)*dxrr[0]+5*xdash*pow(dxrr[0],2)/rr1)+6*cc/pow(rr1,2.5));

dsigma[0][1][1][0] =-Ks*(1-2*ny)/pow(rr1,1.5)+Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[0]-(9-
12*ny)*pow(dxr[1],2)/pow(rr1,2.5)+5*(9-12*ny)*pow(dxr[1],2)*pow(dxrr[0],2)
/pow(rr1,3.5)+30*cc/pow(rr1,3.5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[1],2)-
5*pow(dxr[1],2)*xdash*dxrr[0]/rr1)*dxrr[0]-6*cc/pow(rr1,2.5)*(xdash-5*pow(dxr[1],2)*xdash
/rr1+10*pow(dxr[1],2)*xdash*pow(dxrr[0],2)/pow(rr1,2)))-6*Ks/pow(rr1,2.5)*(xdash*dxrr[0]-(1-
2*ny)*pow(dxr[1],2)-5*pow(dxr[1],2)*xdash*dxrr[0]/rr1);

dsigma[0][2][1][0] =Ks*dxr[1]*dxr[2]*((-9+12*ny)/pow(rr1,2.5)-5*(-9+12*ny)*pow(dxrr[0],2)/pow(rr1,3.5)-
30*cc/pow(rr1,3.5)*(1-2*ny+5*xdash*dxrr[0]/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(5*xdash/rr1-
10*xdash*pow(dxrr[0],2)/pow(rr1,2)))+6*Ks*dxr[1]*dxr[2]/pow(rr1,2.5)*(1-2*ny+5*xdash*dxrr[0]/rr1);
```

```
dsigma[1][1][1][0] =Ks*dxr[1]*(-3*(1-2*ny)*(5-4*ny)/pow(rr1,2.5)*dxrr[0]+5*(9-
12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[0]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(3-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[0]+2*(4-4*ny)*(1-2*ny)
/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*(3-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)
-(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
pow(dxr[1],2)*(3/pow(rr1,0.5)*dxrr[0]+1)/rr1/(pow(rr1,0.5)+dxrr[0])
+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[1],2)*(3*pow(rr
1,0.5)+dxrr[0])/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1))-30*cc/pow(rr1,3.5)*(3*cc-(3-
2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(-3+2*ny-10*pow(dxr[1],2)*xdash
/pow(rr1,2)*dxrr[0]))+Ks*dxr[1]*(6/pow(rr1,2.5)*(3*cc-(3-
2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1)+18*cc/pow(rr1,2.5));

dsigma[1][2][1][0] =Ks*dxr[2]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[0]+5*(9-
12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[0]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)
*(1-pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[0]+2*(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-pow(dxr[1],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-pow(dxr[1],2)
*(3/pow(rr1,0.5)*dxrr[0]+1)/rr1/(pow(rr1,0.5)+dxrr[0])+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/
(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[1],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1))+30*cc*xdash/pow(rr1
,3.5)*(1-5*pow(dxr[1],2)/rr1)*dxrr[0]
-60*cc*xdash/pow(rr1,4.5)*pow(dxr[1],2)*dxrr[0])-6*Ks*dxr[2]*xdash/pow(rr1,2.5)*(1-5*pow(dxr[1],2)/rr1);

dsigma[2][2][1][0] =Ks*dxr[1]*(-3*(1-2*ny)*(3-4*ny)/pow(rr1,2.5)*dxrr[0]+5*(9-
12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[0]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)
*(1-pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[0]+2*(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-pow(dxr[2],2)*
(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-pow(dxr[2],2)
*(3/pow(rr1,0.5)*dxrr[0]+1)/rr1/(pow(rr1,0.5)+dxrr[0])+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/
(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/
pow(rr1,0.5)*dxrr[0]+1))-30*cc/pow(rr1,3.5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(-1+2*ny-
10*pow(dxr[2],2)*xdash/pow(rr1,2)*dxrr[0]))+Ks*dxr[1]*(6/pow(rr1,2.5)*(cc-(1-2*ny)*dxrr[0]+5*pow(dxr[2],2)
*xdash/rr1)+6*cc/pow(rr1,2.5));

dsigma[0][0][2][0] =Ks*dxr[2]*(-3*(-1+2*ny)/pow(rr1,2.5)*dxrr[0]-2*(9-12*ny)*dxrr[0]/pow(rr1,2.5)+5*(9-
12*ny)*pow(dxrr[0],3)/pow(rr1,3.5)-30*cc/pow(rr1,3.5)*(cc+(1-2*ny)*dxrr[0]
```

```
+5*xdash*pow(dxrr[0],2)/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(1-2*ny+10*xdash*dxrr[0]/rr1-
10*xdash*pow(dxrr[0],3)/pow(rr1,2)))+Ks*dxr[2]*(6/pow(rr1,2.5)*(cc+(1-2*ny)*dxrr[0]
+5*xdash*pow(dxrr[0],2)/rr1)+6*cc/pow(rr1,2.5));

dsigma[0][1][2][0] =Ks*dxr[1]*dxr[2]*((-9+12*ny)/pow(rr1,2.5)-5*(-9+12*ny)*pow(dxrr[0],2)/pow(rr1,3.5)-
30*cc/pow(rr1,3.5)*(1-2*ny+5*xdash*dxrr[0]/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(5*xdash/rr1-
10*xdash*pow(dxrr[0],2)/pow(rr1,2)))+6*Ks*dxr[1]*dxr[2]/pow(rr1,2.5)*(1-2*ny+5*xdash*dxrr[0]/rr1);

dsigma[0][2][2][0] =-Ks*(1-2*ny)/pow(rr1,1.5)+Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[0]-(9-
12*ny)*pow(dxr[2],2)/pow(rr1,2.5)+5*(9-12*ny)*pow(dxr[2],2)*pow(dxrr[0],2)
/pow(rr1,3.5)+30*cc/pow(rr1,3.5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[2],2)-
5*pow(dxr[2],2)*xdash*dxrr[0]/rr1)*dxrr[0]-6*cc/pow(rr1,2.5)*(xdash
-5*pow(dxr[2],2)*xdash/rr1+10*pow(dxr[2],2)*xdash*pow(dxrr[0],2)/pow(rr1,2)))-
6*Ks/pow(rr1,2.5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[2],2)-5*pow(dxr[2],2)*xdash*dxrr[0]/rr1);

dsigma[1][1][2][0] =Ks*dxr[2]*(-3*(1-2*ny)*(3-4*ny)/pow(rr1,2.5)*dxrr[0]+5*(9-
12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[0]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[0]+2*(4-4*ny)*(1-2*ny)/pow(rr1,0.5)
/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)-(4-4*ny)*(1-
2*ny)
/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
pow(dxr[1],2)*(3/pow(rr1,0.5)*dxrr[0]+1)/rr1/(pow(rr1,0.5)+dxrr[0])+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0]
)
/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/pow((pow(rr1,0.5)+dxr
r[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1))-30*cc/pow(rr1,3.5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(-1+2*ny-
10*pow(dxr[1],2)*xdash/pow(rr1,2)*dxrr[0]))+Ks*dxr[2]*(6/pow(rr1,2.5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1)+6*cc/pow(rr1,2.5));

dsigma[1][2][2][0] =Ks*dxr[1]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[0]+5*(9-
12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[0]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)
*(1-pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[0]+2*(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),3)
*(1-pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)-(4-
4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(pow(dxr[2],2)*(3/pow(rr1,0.5)*dxrr[0]+1)/rr1/(pow(rr1,0.5
)+dxrr[0])+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]
+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1))+30*cc
*xdash/pow(rr1,3.5)*(1-5*pow(dxr[2],2)/rr1)*dxrr[0]-60*cc*xdash/pow(rr1,4.5)*pow(dxr[2],2)*dxrr[0])-
6*Ks*dxr[1]*xdash/pow(rr1,2.5)*(1-5*pow(dxr[2],2)/rr1);
```

```
dsigma[2][2][2][0] =Ks*dxr[2]*(-3*(1-2*ny)*(5-4*ny)/pow(rr1,2.5)*dxrr[0]+5*(9-
12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[0]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(3-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[0]+2*(4-4*ny)*(1-2*ny)/pow(rr1,0.5)
/pow((pow(rr1,0.5)+dxrr[0]),3)*(3-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*(1/pow(rr1,0.5)*dxrr[0]+1)-(4-4*ny)*(1-
2*ny)
/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
pow(dxr[2],2)*(3/pow(rr1,0.5)*dxrr[0]+1)/rr1/(pow(rr1,0.5)+dxrr[0])+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0]
)
/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[0]+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/pow((pow(rr1,0.5)+dxr
r[0]),2)*(1/pow(rr1,0.5)*dxrr[0]+1))-30*cc/pow(rr1,3.5)*(3*cc-(3-
2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1)*dxrr[0]+6*cc/pow(rr1,2.5)*(-3+2*ny-
10*pow(dxr[2],2)*xdash/pow(rr1,2)*dxrr[0]))+Ks*dxr[2]*(6/pow(rr1,2.5)*(3*cc-(3-
2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1)+18*cc/pow(rr1,2.5));

dsigma[0][0][0][1] =-Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[1]+5*((9-12*ny)*xdash*pow(dxrr[0],2)-
3*cc*dxrr[0]*(5*xdash-cc))/pow(rr1,3.5)*dxrr[1]+210*cc*xdash*pow(dxrr[0],3)/pow(rr1,4.5)*dxrr[1]);

dsigma[0][1][0][1] =-Ks*((1-2*ny)/pow(rr1,1.5)-((9-12*ny)*xdash*dxrr[0]-3*cc*(3*xdash+cc))/pow(rr1,2.5)-
30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5))-Ks*dxr[1]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[1]+5*((9-
12*ny)*xdash*dxrr[0]-
3*cc*(3*xdash+cc))/pow(rr1,3.5)*dxrr[1]+210*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[1]);

dsigma[0][2][0][1] =-Ks*dxr[2]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[1]+5*((9-12*ny)*xdash*dxrr[0]-
3*cc*(3*xdash+cc))/pow(rr1,3.5)*dxrr[1]+210*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[1]);

dsigma[1][1][0][1] =-Ks*(-2*(9-12*ny)*dxr[1]*dxr[0]/pow(rr1,2.5)-60*cc*dxr[1]*xdash*dxrr[0]/pow(rr1,3.5)-
(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(-2*dxr[1]/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-
2*dxr[1]/rr1))-Ks*(-3*(1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr1,2.5)*dxrr[1]+5*((9-
12*ny)*pow(dxr[1],2)*dxr[0]-6*cc*dxrr[0]*((1-2*ny)*xdash-
2*ny*cc))/pow(rr1,3.5)*dxrr[1]+210*cc*pow(dxr[1],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[1]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*(1-pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-
pow(dxr[1],2)
/rr1)*dxrr[1]+(4-4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-pow(dxr[1],2)/rr1)*dxrr[1]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dx
r[1],2)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]+2*pow(dxr[1],2)/pow(rr1,2)*dxrr[1]));

dsigma[1][2][0][1] =-Ks*dxr[2]*((-9+12*ny)*dxr[0]/pow(rr1,2.5)+(4-4*ny)*(1-
2*ny)/rr1/(pow(rr1,0.5)+dxrr[0])*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))
```

```
-30*cc*xdash*dxrr[0]/pow(rr1,3.5))-Ks*dxr[1]*dxr[2]*(-5*(-9+12*ny)*dxr[0]/pow(rr1,3.5)*dxrr[1]-2*(4-
4*ny)*(1-2*ny)/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))*dxrr[1]-(4-
4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))*dxrr[1]+(4-
4*ny)*(1-2*ny)/rr1/(pow(rr1,0.5)+dxrr[0])*(-1/pow((pow(rr1,0.5)+dxrr[0]),2)/pow(rr1,0.5)
*dxrr[1]-1/pow(rr1,1.5)*dxrr[1])+210*cc*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[1]);

dsigma[2][2][0][1] =-Ks*(-3*(1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr1,2.5)*dxrr[1]+5*((9-
12*ny)*pow(dxr[2],2)*dxr[0]-6*cc*dxrr[0]*((1-2*ny)*xdash-
2*ny*cc))/pow(rr1,3.5)*dxrr[1]+210*cc*pow(dxr[2],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[1]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*(1-pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-
pow(dxr[2],2)/rr1)*dxrr[1]+(4-4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)/pow(rr1,0.5)
/(pow(rr1,0.5)+dxrr[0])-pow(dxr[2],2)/rr1)*dxrr[1]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)
+dxrr[0])*dxrr[1]+pow(dxr[2],2)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1]+2*pow(dxr[2],2)/pow(rr1,2)*dxrr[1
]));

dsigma[0][0][1][1] =-Ks*((-1+2*ny)/pow(rr1,1.5)-(9-
12*ny)*pow(dxrr[0],2)/pow(rr1,2.5)+6*cc/pow(rr1,2.5)*(cc+(1-2*ny)*dxrr[0]+5*xdash*pow(dxrr[0],2)/rr1))-
Ks*dxr[1]*(-3*(-1+2*ny)/pow(rr1,2.5)*dxrr[1]+5*(9-12*ny)*pow(dxrr[0],2)/pow(rr1,3.5)*dxrr[1]-
30*cc/pow(rr1,3.5)*(cc+(1-2*ny)*dxrr[0]
+5*xdash*pow(dxrr[0],2)/rr1)*dxrr[1]-60*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[1]);

dsigma[0][1][1][1] =-Ks*(-2*(9-12*ny)*dxr[1]*dxrr[0]/pow(rr1,2.5)-6*cc/pow(rr1,2.5)*(-2*(1-2*ny)*dxr[1]-
10*dxr[1]*xdash*dxrr[0]/rr1))-Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[1]+5*(9-
12*ny)*pow(dxr[1],2)*dxrr[0]/pow(rr1,3.5)*dxrr[1]+30*cc/pow(rr1,3.5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[1],2)-
5*pow(dxr[1],2)*xdash*dxrr[0]/rr1)*dxrr[1]-60*cc*pow(dxr[1],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[1]);

dsigma[0][2][1][1] =-Ks*dxr[2]*((-9+12*ny)*dxrr[0]/pow(rr1,2.5)+6*cc/pow(rr1,2.5)*(1-
2*ny+5*xdash*dxrr[0]/rr1))-Ks*dxr[1]*dxr[2]*(-5*(-9+12*ny)*dxrr[0]/pow(rr1,3.5)*dxrr[1]-
30*cc/pow(rr1,3.5)*(1-2*ny+5*xdash*dxrr[0]/rr1)*dxrr[1]-60*cc*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[1]);

dsigma[1][1][1][1] =-Ks*((1-2*ny)*(5-4*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[1],2)/pow(rr1,2.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(3-pow(dxr[1],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))+6*cc/pow(rr1,2.5)*(3*cc-(3-
2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1))-Ks*dxr[1]*(-2*(9-12*ny)*dxr[1]/pow(rr1,2.5)+2*(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*dxr[1]*(3*pow(rr1,0.5)+dxrr[0])+60*cc/pow(rr1,3.5)*dxr[1]*
xdash)
-Ks*dxr[1]*(-3*(1-2*ny)*(5-4*ny)/pow(rr1,2.5)*dxrr[1]+5*(9-12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[1]+(4-
4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(3-
```

```
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)
*(3-pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr
1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+d
xrr[0]),2)*dxrr[1])-30*cc/pow(rr1,3.5)*(3*cc-(3-2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash
/rr1)*dxrr[1]-60*cc/pow(rr1,4.5)*pow(dxr[1],2)*xdash*dxrr[1]);

dsigma[1][2][1][1] =-Ks*dxr[2]*(-2*(9-12*ny)*dxr[1]/pow(rr1,2.5)+2*(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*dxr[1]*(3*pow(rr1,0.5)+dxrr[0])+60*cc/pow(rr1,3.5)*dxr[1]*
xdash)-Ks*dxr[2]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[1]+5*(9-12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[1]+(4-
4*ny)*(1-2*ny)
/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]+2*(4-4*ny)*(1-2*ny)
/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr
1,2)
/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[
0]),2)*dxrr[1])+30*cc*xdash/pow(rr1,3.5)*(1-5*pow(dxr[1],2)/rr1)*dxrr[1]-
60*cc/pow(rr1,4.5)*pow(dxr[1],2)*xdash*dxrr[1]);

dsigma[2][2][1][1] =-Ks*((1-2*ny)*(3-4*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[2],2)/pow(rr1,2.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-pow(dxr[2],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))+6*cc/pow(rr1,2.5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1))-Ks*dxr[1]*(-3*(1-2*ny)*(3-4*ny)/pow(rr1,2.5)*dxrr[1]+5*(9-
12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[1]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])
/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]-(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]
+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dxr[2],2)*(3*pow(rr
1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[1])-30*cc/pow(rr1,3.5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1)*dxrr[1]-60*cc/pow(rr1,4.5)*pow(dxr[2],2)*xdash*dxrr[1]);

dsigma[0][0][2][1] =-Ks*dxr[2]*(-3*(-1+2*ny)/pow(rr1,2.5)*dxrr[1]+5*(9-
12*ny)*pow(dxrr[0],2)/pow(rr1,3.5)*dxrr[1]-30*cc/pow(rr1,3.5)*(cc+(1-2*ny)*dxrr[0]
+5*xdash*pow(dxrr[0],2)/rr1)*dxrr[1]-60*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[1]);
```

```
dsigma[0][1][2][1] =-Ks*dxr[2]*((-9+12*ny)*dxrr[0]/pow(rr1,2.5)+6*cc/pow(rr1,2.5)*(1-
2*ny+5*xdash*dxrr[0]/rr1))-Ks*dxr[1]*dxr[2]*(-5*(-9+12*ny)*dxrr[0]/pow(rr1,3.5)*dxrr[1]-
30*cc/pow(rr1,3.5)*(1-2*ny+5*xdash*dxrr[0]/rr1)*dxrr[1]-60*cc*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[1]);

dsigma[0][2][2][1] =-Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[1]+5*(9-
12*ny)*pow(dxr[2],2)*dxrr[0]/pow(rr1,3.5)*dxrr[1]+30*cc/pow(rr1,3.5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[2],2)-
5*pow(dxr[2],2)*xdash*dxrr[0]/rr1)*dxrr[1]-60*cc*pow(dxr[2],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[1]);

dsigma[1][1][2][1] =-Ks*dxr[2]*(-2*(9-12*ny)*dxr[1]/pow(rr1,2.5)+2*(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*dxr[1]*(3*pow(rr1,0.5)+dxrr[0])+60*cc/pow(rr1,3.5)*dxr[1]*
xdash)-Ks*dxr[2]*(-3*(1-2*ny)*(3-4*ny)/pow(rr1,2.5)*dxrr[1]+5*(9-12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[1]
+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])
/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1
,0.5)+dxrr[0]),2)*dxrr[1])-30*cc/pow(rr1,3.5)*(cc-(1-2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1)*dxrr[1]-
60*cc/pow(rr1,4.5)*pow(dxr[1],2)*xdash*dxrr[1]);

dsigma[1][2][2][1] =-Ks*((1-2*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[2],2)/pow(rr1,2.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))-6*cc*xdash/pow(rr1,2.5)*(1-
5*pow(dxr[2],2)/rr1))-Ks*dxr[1]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[1]
+5*(9-12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[1]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])
/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]-(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr
1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+d
xrr[0]),2)*dxrr[1])+30*cc*xdash/pow(rr1,3.5)*(1-5*pow(dxr[2],2)/rr1)*dxrr[1]-
60*cc/pow(rr1,4.5)*pow(dxr[2],2)*xdash*dxrr[1]);

dsigma[2][2][2][1] =-Ks*dxr[2]*(-3*(1-2*ny)*(5-4*ny)/pow(rr1,2.5)*dxrr[1]+5*(9-
12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[1]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)
*(3-pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(3-pow(dxr[2],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[1]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-3*pow(dxr[2],2)
```

```
/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[1]+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/(pow(rr1,0
.5)+dxrr[0])*dxrr[1]+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr
[1])-30*cc/pow(rr1,3.5)*(3*cc-(3-2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1)*dxrr[1]-
60*cc/pow(rr1,4.5)*pow(dxr[2],2)*xdash*dxrr[1]);

dsigma[0][0][0][2] =-Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[2]+5*((9-12*ny)*xdash*pow(dxrr[0],2)-
3*cc*dxrr[0]*(5*xdash-cc))/pow(rr1,3.5)*dxrr[2]+210*cc*xdash*pow(dxrr[0],3)/pow(rr1,4.5)*dxrr[2]);

dsigma[0][1][0][2] =-Ks*dxr[1]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[2]+5*((9-12*ny)*xdash*dxrr[0]-
3*cc*(3*xdash+cc))/pow(rr1,3.5)*dxrr[2]+210*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[2]);

dsigma[0][2][0][2] =-Ks*((1-2*ny)/pow(rr1,1.5)-((9-12*ny)*xdash*dxrr[0]-3*cc*(3*xdash+cc))/pow(rr1,2.5)-
30*cc*xdash*pow(dxrr[0],2)/pow(rr1,3.5))-Ks*dxr[2]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[2]+5*((9-
12*ny)*xdash*dxrr[0]-
3*cc*(3*xdash+cc))/pow(rr1,3.5)*dxrr[2]+210*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[2]);

dsigma[1][1][0][2] =-Ks*(-3*(1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr1,2.5)*dxrr[2]+5*((9-
12*ny)*pow(dxr[1],2)*dxr[0]-6*cc*dxrr[0]*((1-2*ny)*xdash-2*ny*cc))
/pow(rr1,3.5)*dxrr[2]+210*cc*pow(dxr[1],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[2]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*(1-pow(dxr[1],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-
pow(dxr[1],2)/rr1)*dxrr[2]+(4-4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-pow(dxr[1],2)
/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-pow(dxr[1],2)/rr1)*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+pow(dx
r[1],2)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]+2*pow(dxr[1],2)/pow(rr1,2)*dxrr[2]));

dsigma[1][2][0][2] =-Ks*dxr[1]*((-9+12*ny)*dxr[0]/pow(rr1,2.5)+(4-4*ny)*(1-
2*ny)/rr1/(pow(rr1,0.5)+dxrr[0])*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))-
30*cc*xdash*dxrr[0]/pow(rr1,3.5))-Ks*dxr[1]*dxr[2]*(-5*(-9+12*ny)*dxr[0]/pow(rr1,3.5)*dxrr[2]-2*(4-
4*ny)*(1-2*ny)/pow(rr1,2)/
(pow(rr1,0.5)+dxrr[0])*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1/(pow(rr1,0.5)+dxrr[0])+1/pow(rr1,0.5))*dxrr[2]+(4-
4*ny)*(1-2*ny)/rr1/(pow(rr1,0.5)+dxrr[0])*(-1/pow((pow(rr1,0.5)+dxrr[0]),2)/pow(rr1,0.5)*dxrr[2]
-1/pow(rr1,1.5)*dxrr[2])+210*cc*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[2]);

dsigma[2][2][0][2] =-Ks*(-2*(9-12*ny)*dxr[2]*dxr[0]/pow(rr1,2.5)-60*cc*dxr[2]*xdash*dxrr[0]/pow(rr1,3.5)-
(4-4*ny)*(1-2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(-2*dxr[2]/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-
2*dxr[2]/rr1))-Ks*(-3*(1-2*ny)*(3*dxr[0]-4*ny*dxrr[0])/pow(rr1,2.5)*dxrr[2]+5*((9-
12*ny)*pow(dxr[2],2)*dxr[0]-6*cc*dxrr[0]*((1-2*ny)*xdash-
2*ny*cc))/pow(rr1,3.5)*dxrr[2]+210*cc*pow(dxr[2],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[2]
+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*(1-
pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-pow(dxr[2],2)/rr1)*dxrr[2]+(4-4*ny)*(1-
```

```
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-pow(dxr[2],2)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])-
pow(dxr[2],2)/rr1)*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/(pow(rr1,0.5)+dxrr[0])*(pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+pow(dx
r[2],2)/rr1/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2]+2*pow(dxr[2],2)/pow(rr1,2)*dxrr[2]));

dsigma[0][0][1][2] =-Ks*dxr[1]*(-3*(-1+2*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-
12*ny)*pow(dxrr[0],2)/pow(rr1,3.5)*dxrr[2]-30*cc/pow(rr1,3.5)*(cc+(1-
2*ny)*dxrr[0]+5*xdash*pow(dxrr[0],2)/rr1)*dxrr[2]-60*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[2]);

dsigma[0][1][1][2] =-Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[2]+5*(9-
12*ny)*pow(dxr[1],2)*dxrr[0]/pow(rr1,3.5)*dxrr[2]+30*cc/pow(rr1,3.5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[1],2)-
5*pow(dxr[1],2)*xdash*dxrr[0]/rr1)*dxrr[2]-60*cc*pow(dxr[1],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[2]);

dsigma[0][2][1][2] =-Ks*dxr[1]*((-9+12*ny)*dxrr[0]/pow(rr1,2.5)+6*cc/pow(rr1,2.5)*(1-
2*ny+5*xdash*dxrr[0]/rr1))-Ks*dxr[1]*dxr[2]*(-5*(-9+12*ny)*dxrr[0]/pow(rr1,3.5)*dxrr[2]-
30*cc/pow(rr1,3.5)*(1-2*ny+5*xdash*dxrr[0]/rr1)*dxrr[2]-60*cc*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[2]);

dsigma[1][1][1][2] =-Ks*dxr[1]*(-3*(1-2*ny)*(5-4*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-
12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[2]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(3-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]+2*(4-4*ny)*(1-2*ny)/rr1/
pow((pow(rr1,0.5)+dxrr[0]),3)*(3-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[1],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr
1,2)
/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[
0]),2)*dxrr[2])-30*cc/pow(rr1,3.5)*(3*cc-(3-2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1)*dxrr[2]-
60*cc/pow(rr1,4.5)*pow(dxr[1],2)*xdash*dxrr[2]);

dsigma[1][2][1][2] =-Ks*((1-2*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[1],2)/pow(rr1,2.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-pow(dxr[1],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))-6*cc*xdash/pow(rr1,2.5)*(1-5*pow(dxr[1],2)/rr1))-
Ks*dxr[2]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[2]+(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-pow(dxr[1],2)*(3*
pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-pow(dxr[1],2)*(3*
pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-3*pow(dxr[1],2)
/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/(pow(rr1,0
.5)+dxrr[0])*dxrr[2]+pow(dxr[1],2)
```

```
*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr[2])+30*cc*xdash/pow(rr1,3.5)*(1-
5*pow(dxr[1],2)/rr1)*dxrr[2]-60*cc/pow(rr1,4.5)*pow(dxr[1],2)*xdash*dxrr[2]);

dsigma[2][2][1][2] =-Ks*dxr[1]*(-2*(9-12*ny)*dxr[2]/pow(rr1,2.5)+2*(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*dxr[2]*(3*pow(rr1,0.5)
+dxrr[0])+60*cc/pow(rr1,3.5)*dxr[2]*xdash)-Ks*dxr[1]*(-3*(1-2*ny)*(3-4*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-
12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[2]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]
+2*(4-4*ny)*(1-2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]-(4-4*ny)*(1-2*ny)
/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])
/pow(rr1,2)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1
,0.5)+dxrr[0]),2)*dxrr[2])-30*cc/pow(rr1,3.5)*(cc-(1-2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1)*dxrr[2]-
60*cc/pow(rr1,4.5)*pow(dxr[2],2)*xdash*dxrr[2]);

dsigma[0][0][2][2] =-Ks*((-1+2*ny)/pow(rr1,1.5)-(9-
12*ny)*pow(dxrr[0],2)/pow(rr1,2.5)+6*cc/pow(rr1,2.5)*(cc+(1-2*ny)*dxrr[0]+5*xdash*pow(dxrr[0],2)/rr1))-
Ks*dxr[2]*(-3*(-1+2*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-12*ny)*pow(dxrr[0],2)/pow(rr1,3.5)*dxrr[2]-
30*cc/pow(rr1,3.5)*(cc+(1-2*ny)*dxrr[0]
+5*xdash*pow(dxrr[0],2)/rr1)*dxrr[2]-60*cc*xdash*pow(dxrr[0],2)/pow(rr1,4.5)*dxrr[2]);

dsigma[0][1][2][2] =-Ks*dxr[1]*((-9+12*ny)*dxrr[0]/pow(rr1,2.5)+6*cc/pow(rr1,2.5)*(1-
2*ny+5*xdash*dxrr[0]/rr1))-Ks*dxr[1]*dxr[2]*(-5*(-9+12*ny)*dxrr[0]/pow(rr1,3.5)*dxrr[2]-
30*cc/pow(rr1,3.5)*(1-2*ny+5*xdash*dxrr[0]/rr1)*dxrr[2]-60*cc*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[2]);

dsigma[0][2][2][2] =-Ks*(-2*(9-12*ny)*dxr[2]*dxrr[0]/pow(rr1,2.5)-6*cc/pow(rr1,2.5)*(-2*(1-2*ny)*dxr[2]-
10*dxr[2]*xdash*dxrr[0]/rr1))-Ks*(-3*(1-2*ny)*dxr[0]/pow(rr1,2.5)*dxrr[2]+5*(9-
12*ny)*pow(dxr[2],2)*dxrr[0]/pow(rr1,3.5)*dxrr[2]+30*cc/pow(rr1,3.5)*(xdash*dxrr[0]-(1-2*ny)*pow(dxr[2],2)-
5*pow(dxr[2],2)*xdash*dxrr[0]/rr1)*dxrr[2]-60*cc*pow(dxr[2],2)*xdash*dxrr[0]/pow(rr1,4.5)*dxrr[2]);

dsigma[1][1][2][2] =-Ks*((1-2*ny)*(3-4*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[1],2)/pow(rr1,2.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))+6*cc/pow(rr1,2.5)*(cc-(1-
2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1))-Ks*dxr[2]*(-3*(1-2*ny)*(3-4*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-
12*ny)*pow(dxr[1],2)/pow(rr1,3.5)*dxrr[2]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)
*(1-pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-pow(dxr[1],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-3*pow(dxr[1],2)
```

```
/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+2*pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/(pow(rr1,0
.5)+dxrr[0])*dxrr[2]+pow(dxr[1],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr
[2])-30*cc/pow(rr1,3.5)*(cc-(1-2*ny)*dxrr[0]+5*pow(dxr[1],2)*xdash/rr1)*dxrr[2]-
60*cc/pow(rr1,4.5)*pow(dxr[1],2)*xdash*dxrr[2]);

dsigma[1][2][2][2] =-Ks*dxr[1]*(-2*(9-12*ny)*dxr[2]/pow(rr1,2.5)+2*(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*dxr[2]*(3*pow(rr1,0.5)+dxrr[0])+60*cc/pow(rr1,3.5)*dxr[2]*
xdash)-Ks*dxr[1]*(-3*(1-2*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[2]+(4-
4*ny)*(1-2*ny)
/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(1-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]+2*(4-4*ny)*(1-2*ny)
/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(1-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-
3*pow(dxr[2],2)/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr
1,2)/
(pow(rr1,0.5)+dxrr[0])*dxrr[2]+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0
]),2)*dxrr[2])+30*cc*xdash/pow(rr1,3.5)*(1-5*pow(dxr[2],2)/rr1)*dxrr[2]-
60*cc/pow(rr1,4.5)*pow(dxr[2],2)*xdash*dxrr[2]);

dsigma[2][2][2][2] =-Ks*((1-2*ny)*(5-4*ny)/pow(rr1,1.5)-(9-12*ny)*pow(dxr[2],2)/pow(rr1,2.5)-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(3-pow(dxr[2],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))+6*cc/pow(rr1,2.5)*(3*cc-(3-
2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1))-Ks*dxr[2]*(-2*(9-12*ny)*dxr[2]/pow(rr1,2.5)+2*(4-4*ny)*(1-
2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),3)*dxr[2]*(3*pow(rr1,0.5)+dxrr[0])+60*cc/pow(rr1,3.5)*dxr[2]*
xdash)-Ks*dxr[2]*(-3*(1-2*ny)*(5-4*ny)/pow(rr1,2.5)*dxrr[2]+5*(9-
12*ny)*pow(dxr[2],2)/pow(rr1,3.5)*dxrr[2]+(4-4*ny)*(1-2*ny)/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(3-
pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]+2*(4-4*ny)*(1-
2*ny)/rr1/pow((pow(rr1,0.5)+dxrr[0]),3)*(3-pow(dxr[2],2)
*(3*pow(rr1,0.5)+dxrr[0])/rr1/(pow(rr1,0.5)+dxrr[0]))*dxrr[2]-(4-4*ny)*(1-
2*ny)/pow(rr1,0.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*(-3*pow(dxr[2],2)
/pow(rr1,1.5)/(pow(rr1,0.5)+dxrr[0])*dxrr[2]+2*pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,2)/(pow(rr1,0
.5)+dxrr[0])*dxrr[2]+pow(dxr[2],2)*(3*pow(rr1,0.5)+dxrr[0])/pow(rr1,1.5)/pow((pow(rr1,0.5)+dxrr[0]),2)*dxrr
[2])-30*cc/pow(rr1,3.5)*(3*cc-(3-2*ny)*dxrr[0]+5*pow(dxr[2],2)*xdash/rr1)*dxrr[2]-
60*cc/pow(rr1,4.5)*pow(dxr[2],2)*xdash*dxrr[2]);

dsigma[1][0][0][0]=dsigma[0][1][0][0];
dsigma[2][0][0][0]=dsigma[0][2][0][0];
dsigma[2][1][0][0]=dsigma[1][2][0][0];
dsigma[1][0][1][0]=dsigma[0][1][1][0];
dsigma[2][0][1][0]=dsigma[0][2][1][0];
```

```
dsigma[2][1][1][0]=dsigma[1][2][1][0];
dsigma[1][0][2][0]=dsigma[0][1][2][0];
dsigma[2][0][2][0]=dsigma[0][2][2][0];
dsigma[2][1][2][0]=dsigma[1][2][2][0];

dsigma[1][0][0][1]=dsigma[0][1][0][1];
dsigma[2][0][0][1]=dsigma[0][2][0][1];
dsigma[2][1][0][1]=dsigma[1][2][0][1];
dsigma[1][0][1][1]=dsigma[0][1][1][1];
dsigma[2][0][1][1]=dsigma[0][2][1][1];
dsigma[2][1][1][1]=dsigma[1][2][1][1];
dsigma[1][0][2][1]=dsigma[0][1][2][1];
dsigma[2][0][2][1]=dsigma[0][2][2][1];
dsigma[2][1][2][1]=dsigma[1][2][2][1];

dsigma[1][0][0][2]=dsigma[0][1][0][2];
dsigma[2][0][0][2]=dsigma[0][2][0][2];
dsigma[2][1][0][2]=dsigma[1][2][0][2];
dsigma[1][0][1][2]=dsigma[0][1][1][2];
dsigma[2][0][1][2]=dsigma[0][2][1][2];
dsigma[2][1][1][2]=dsigma[1][2][1][2];
dsigma[1][0][2][2]=dsigma[0][1][2][2];
dsigma[2][0][2][2]=dsigma[0][2][2][2];
dsigma[2][1][2][2]=dsigma[1][2][2][2];
 for (int i = 0; i < Cdim; ++i)
    for (int j = 0; j < Cdim; ++j)
     for (int k = 0; k < Cdim; ++k)
  {
       USMindlin1[i][j][k] = 0;
  }
for (int i = 0; i < Cdim; ++i)
 for (int j = 0; j < Cdim; ++j)
  for (int k = 0; k < Cdim; ++k)
   for (int m = 0; m < Cdim; ++m)
    {
        if (i==j)
         {
             delta=1;
         }
        else
        {
```

```
            delta=0;
         }

USMindlin1[i][j][k]=USMindlin1[i][j][k]+G*(dsigma[k][m][i][j]+dsigma[k][m][j][i]+C*delta*(dsigma[k][m][0][0
]+dsigma[k][m][1][1]+dsigma[k][m][2][2]))*Vnor[m];
     }
USMindlin[0][0] =USMindlin1[0][0][0];
USMindlin[0][1] =USMindlin1[0][0][1];
USMindlin[0][2] =USMindlin1[0][0][2];
USMindlin[1][0] =USMindlin1[1][1][0];
USMindlin[1][1] =USMindlin1[1][1][1];
USMindlin[1][2] =USMindlin1[1][1][2];
USMindlin[2][0] =USMindlin1[2][2][0];
USMindlin[2][1] =USMindlin1[2][2][1];
USMindlin[2][2] =USMindlin1[2][2][2];
USMindlin[3][0] =USMindlin1[0][1][0];
USMindlin[3][1] =USMindlin1[0][1][1];
USMindlin[3][2] =USMindlin1[0][1][2];
USMindlin[4][0] =USMindlin1[1][2][0];
USMindlin[4][1] =USMindlin1[1][2][1];
USMindlin[4][2] =USMindlin1[1][2][2];
USMindlin[5][0] =USMindlin1[0][2][0];
USMindlin[5][1] =USMindlin1[0][2][1];
USMindlin[5][2] =USMindlin1[0][2][2];

return;
```

7. Appendix g

```
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
; Numerical solution for a spherical Excavation in Elastic material
;     (Theoretical solution by R. V. Southwell (1926) problem
;                        //Uncoupled Solution//
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
New
CONFIG cppfish
LOAD function BEMKMM002_64.dll
def testt
tim0=clock
end
def test1
  tim=(clock-tim0)/100.0
end
@testt
def parm
 rad=1.0         ; radius of spherical cavity
 len= 2          ; length of outer box edge
 in_size= 5      ; number of zones in radial direction
 rad_size= 8     ; number of zones along outer cube edge
 global vx = 0.0
 global vy = 0.0
 global vz = 0.0
end
@parm
gen zone radbrick edge @len size @in_size @in_size @in_size @rad_size &
rat 1.4 1.4 1.4 1.0 dim @rad @rad @rad
def make_sphere
 ; Loop over all GPs and remap their coordinates:
 ; assume len >rad
 p_gp=gp_head
 loop while p_gp#null
  ; Get gp coordinate: P=(px,py,pz)
  px=gp_xpos(p_gp)
  py=gp_ypos(p_gp)
  pz=gp_zpos(p_gp)
  ; Compute A=(ax,ay,az)=point on sphere radially "below" P.
```

```
   dist=sqrt(px*px+py*py+pz*pz)
   if dist>0 then
    k=rad/dist
    ax=px*k
    ay=py*k
    az=pz*k
    ; Compute B=(bx,by,bz)=point on outer box boundary radially "above" P.
    maxp=max(px,max(py,pz))
    k=len/dist
    bx=px*k
    by=py*k
    bz=pz*k
    ; Linear interpolation: P=A+u*(B-A)
    u=(maxp-rad)/(len-rad)
    gp_xpos(p_gp)=ax+u*(bx-ax)
    gp_ypos(p_gp)=ay+u*(by-ay)
    gp_zpos(p_gp)=az+u*(bz-az)
   end_if
   p_gp=gp_next(p_gp)
 end_loop
 xinitialstress=0
 yinitialstress=0
 zinitialstress=-1.0
 E_Young = 1000.0                ; Young Modulus
 ny_Poisson = 0.0                ; Poisson's Ratio
 Bulk_modulus=E_Young/(3*(1-2*ny_Poisson))
 G_kelvin=E_Young/(2*(1+ny_Poisson))
 zappi=len+0.001
 zappf=-len-0.001
 radi=rad - 0.1
 radf=rad + 0.1
end
@make_sphere
;gen zone reflect dip 0 dd 90 origin 0 0 0               ;If symmetry about xz and yz only;
plot create view Spherical-excavation
plot set name 'Spherical excavation'
plot add zone addlabel "Zone" yellow
plot boundary
plot add axes
plot show
model me elastic
```

```
prop bulk @Bulk_modulus shear @G_kelvin
ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress
apply szz=@zinitialstress range z 0.001,@zappi
;apply szz=@zinitialstress range z @zappi @zappf                                    ;If symmetry about xz and yz
only;

apply remove szz range ann c 0.0 0.0 0.0 r @radi @radf
apply xv 0 range x -0.01 0.01
apply yv 0 range y -0.01 0.01
apply zv 0 range z -0.01 0.01
set mech rat 1.e-6
hist n 5
hist add unbal
hist add gp zdisp 0 0 1
plot history 2
def totalstepnumber1
 Whilestepping
 STEPNUMBER1=STEPNUMBER1+1
end
solve
def totalstepnumber
 STEPNUMBER=STEPNUMBER1
end
@totalstepnumber
;-------------------------------------------------------------------------
; exact and numerical solution for a spherical Excavation problem
;
; stores in
; Table 1: analytical values sigzze
; Table 2: numerical values sigzz
; at zone centroid closed to x axis
;
; Table 3: errzr the error in sigzz
;
; stores in
; Table 4: numerical values of ux at grid points
;-------------------------------------------------------------------------
def nastr
 rad = 1
 command
  tab 1 name analytic-sigzz
```

```
  tab 2 name FLAC3D -sigzz
  tab 3 name ERR-in-Sigzz
 end_command
 zpnt = zone_head
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   if gp_zpos(gpnt) < 0.001 then
   ;if gp_zpos(gpnt) > -0.001 then                  ;If symmetry about xz and yz only;
    if gp_ypos(gpnt) < 0.001 then
    ; if gp_ypos(gpnt) > -0.001 then                ;If symmetry about xz and yz only;
      mark = 1
    ;end_if
    end_if
   ;end_if
   end_if
  end_loop
  radd= sqrt(z_xcen(zpnt)^2 +z_ycen(zpnt)^2+z_zcen(zpnt)^2)
  A=(4-ny_Poisson *5)*(rad^3)
  B=2*(7-ny_Poisson *5)*(radd^3)
  D=4.5*(rad^5)
  E=(7-5*ny_Poisson )*(radd^5)
  sigzze = zinitialstress*(1.+(A/B)+(D/E))
  sigzz = z_szz(zpnt)
  errzr = ((sigzz - sigzze)/sigzze)*100
  if mark = 1
   table(1,radd) = sigzze
   table(2,radd) = sigzz
   table(3,radd) = errzr
  end_if
  zpnt = z_next(zpnt)
 end_loop
 Tot_Zone= nzone
end

def nadis
 command
  tab 4 name FLAC3D -ux
 end_command
 pnt = gp_head
```

```
 loop while pnt # null
  mark = 0
  if gp_zpos(pnt) < 0.001 then
  ; if gp_zpos(pnt) > -0.001 then                    ;If symmetry about xz and yz only;
   if gp_ypos(pnt) < 0.001 then
  ; if gp_ypos(pnt) > -0.001 then                    ;If symmetry about xz and yz only;
    mark = 1
   end_if
  ;end_if
  end_if
  ;end_if
  radd= sqrt(gp_xpos(pnt)^2 +gp_ypos(pnt)^2+ gp_zpos(pnt)^2)
  dis = gp_xdisp(pnt)
  if mark = 1 then
   table(4,radd) =dis
  end_if
  pnt = gp_next(pnt)
 end_loop
 gpnt = gp_near(0,0,rad)
 diss = gp_zdisp(gpnt)
end
@nastr
@nadis
@test1
list @tim
plot create view verticalstresses
plot add table 1 , 2
plot create view displacements
plot add table 4
pause
```

```
;-------------------------------------------------------------------------------------------------
; Numerical solution for a surface uniform load in semi-infinite Elastic medium {Holl Solution}
;                                  //Uncoupled Solution//
;-------------------------------------------------------------------------------------------------
New
CONFIG cppfish
LOAD function BEMKMM002_64.dll
def testt
 tim0=clock
end
def test1
 tim=(clock-tim0)/100.0
end
@testt
def parm
 global d = 3.0   ; length of outer box edge
 global len1 = d
 global len2 = -d
 global len3 = -d
 in_size1= 8      ; number of zones along outer cube edge
 in_size2= 8
 in_size3= 8
 in_size4= 8
 xinitialstress=0.0
 yinitialstress=0.0
 zinitialstress=0.0
 global E_Young=10000           ; Young Modulus
 global ny_Poisson = 0.0        ; Poisson's Ratio
 global Bulk_modulus=E_Young/(3*(1-2*ny_Poisson))
 global G_kelvin=E_Young/(2*(1+ny_Poisson))
 global YLength = -1
 global XLength = 1
 global rad = XLength
 global YLength2 =-YLength
 global XLength1 =2*XLength
 global YLength1 =-2*YLength
 global Load = -100.00
end
@parm
```

```
gen zone radbrick size @in_size1 @in_size2 @in_size3 @in_size4 rat 1.0 1.0 1.0 1 p0 0 0 0 p1 @len1 0 0 p2 0
@len2 & 0 p3 0 0 @len3 p4 @len1 @len2 0 p5 0 @len2 @len3 p6 @len1 0 @len3 p7 @len1 @len2 @len3 dim @XLength
@YLength2 & @YLength2
def make_sphere
 ; Loop over all GPs and remap their coordinates:
 ; assume len > rad
 p_gp=gp_head
 loop while p_gp#null
  ; Get gp coordinate: P=(px,py,pz)
  px=gp_xpos(p_gp)
  py=gp_ypos(p_gp)
  pz=gp_zpos(p_gp)
  maxp=max(abs(px),max(abs(py),abs(pz)))
  ; Compute A=(ax,ay,az)=point on sphere radially "below" P.
  dist=sqrt(px*px+py*py+pz*pz)
  if dist> 0 then
   k=rad/maxp
   ax=px*k
   ay=py*k
   az=pz*k
   ; Compute B=(bx,by,bz)=point on outer box boundary radially "above" P.
   k=d/dist
   bx=px*k
   by=py*k
   bz=pz*k
   ; Linear interpolation: P=A+u*(B-A)
   u=(maxp-rad)/(d-rad)
   gp_xpos(p_gp)=ax+u*(bx-ax)
   gp_ypos(p_gp)=ay+u*(by-ay)
   gp_zpos(p_gp)=az+u*(bz-az)
  end_if
  p_gp=gp_next(p_gp)
 end_loop
end
@make_sphere
gen zone brick size @in_size1 @in_size2 @in_size3 rat 1.0 1.0 1.0 p0 0 0 0 p1 @XLength 0 0 p2 0 @YLength 0
p3 0 0 @YLength
; gen zone reflect dip 90 dd 90 origin 0 0 0
; gen zone reflect dip 90 dd 180 origin 0 0 0          ;if No symmetry;
plot create view Brick
plot set name 'Brick'
```

```
plot add zone addlabel "Zone" yellow
plot boundary
plot add axes
model me elastic
prop bulk @Bulk_modulus shear @G_kelvin
ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress
apply szz @Load range x 0 ,@XLength y 0, @YLength z 0.001 -0.001
apply xv 0 range x -0.0001 0.0001
apply yv 0 range y -0.0001 0.0001
;apply szz @Load range x @YLength ,@XLength y @XLength, @YLength z 0.001 -0.001    ; If No symmetry;
def fixboundary
 p_gp =gp_head
 loop while p_gp # null
  distance = sqrt(gp_xpos(p_gp)^2+gp_ypos(p_gp)^2+gp_zpos(p_gp)^2)
  dv=gp_id(p_gp)
  if distance >= d-0.001 then
   COMMAND
    Apply zv 0 range id @dv
   END_COMMAND
  end_if
  p_gp=gp_next(p_gp)
 end_loop
end
@fixboundary
set mech rat 1.e-6
hist n 5
hist add unbal
hist add gp zdisp 0 0 0
hist add gp zdisp @XLength @YLength 0
def totalstepnumber1
 Whilestepping
 STEPNUMBER1=STEPNUMBER1+1
end
solve
def totalstepnumber
 STEPNUMBER=STEPNUMBER1
end
@totalstepnumber
;-----------------------------------------------------------------------------------------
; exact and numerical solution for a uniform surface load in Semi-infinite medium
; Table 1: analytical values sigzze
```

```
; Table 2: numerical values sigzz
; Table 4: analytical values sigxxe
; Table 5: numerical values sigxx
; Table 7: analytical values sigxze
; Table 8: numerical values sigxz
; at zone centroid under the corner of the load area
;-------------------------------------------------------------------------------------
def nastr
 command
  table 1 name Analytic-Corner-Sigzze
  table 2 name FLAC3D -Corner-Sigzz
  table 3 name Error-Corner-Sigzz
  table 4 name Analytic-Corner-Sigxxe
  table 5 name FLAC3D -Corner-Sigxx
  table 6 name Error-Corner-Sigxx
  table 7 name Analytic-Corner-Sigxze
  table 8 name FLAC3D -Corner-Sigxz
  table 9 name Error-Corner-Sigxz
 end_command
 Num = 0
 zpnt = zone_head
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   if gp_xpos(gpnt) = XLength then
    if gp_ypos(gpnt) = YLength then
     if z_xcen(zpnt)< XLength then
      if z_ycen(zpnt)> YLength then
        mark = 1
      end_if
     end_if
    end_if
   end_if
  end_loop
  if mark = 1 then
   local Depth =abs(z_zcen(zpnt))
   local R1=sqrt(XLength1^2+Depth^2)
   local R2=sqrt(YLength1^2+Depth^2)
   local R3=sqrt(XLength1^2+YLength1^2+Depth^2)
```

```
    local
    Iz4=(1/(2*pi))*(ATAN((XLength1*YLength1)/(Depth*R3))+((XLength1*YLength1*Depth/R3)*((1/R1^2)+(1/R2^2))))
    local Ix4=(1/(4*pi))*(ATAN((XLength1*YLength1)/(Depth*R3))-((2*XLength1*YLength1*Depth)/(R1^2*R3)))
    local Ixz4=(-1/(2*pi))*(((XLength1)/(R2))-((XLength1*Depth^2)/(R1^2*R3)))
    sigzze= -Iz4*Load
    sigxxe= -Ix4*Load
    sigxze= Ixz4*Load
    sigzz = -z_szz(zpnt)
    sigxx = -z_sxx(zpnt)
    sigxz = z_sxz(zpnt)
    errz_corner = ((sigzz - sigzze)/abs(sigzze))*100
    errx_corner = ((sigxx - sigxxe)/abs(sigxxe))*100
    errxz_corner = ((sigxz - sigxze)/abs(sigxze))*100
    Num = Num +1
    table(1,sigzze) = -Depth
    table(2,sigzz) = -Depth
    table(3,errz_corner) = -Depth
    table(4,sigxxe) = -Depth
    table(5,sigxx) = -Depth
    table(6,errx_corner) = -Depth
    table(7,sigxze) = -Depth
    table(8,-Depth) = sigxz
    table(9,errxz_corner) = -Depth
   end_if
   zpnt = z_next(zpnt)
  end_loop
  loop i(1,Num)
   local x1 = xtable(8,i)
   local y1 = ytable(8,i)
   xtable(8,i) = y1
   ytable(8,i) = x1
  end_loop
end
;-------------------------------------------------------------------------------------
; exact and numerical solution for a uniform surface load in Semi-infinite medium
; Table 10: analytical values sigzze
; Table 11: numerical values sigzz
; Table 13: analytical values sigxxe
; Table 14: numerical values sigxx
; at zone centroid under the center of the load area
;-------------------------------------------------------------------------------------
```

```
def nastr1
 command
  table 10 name Analytic-Center-Sigzze
  table 11 name FLAC3D -Center-Sigzz
  table 12 name Error-Center-Sigzz
  table 13 name Analytic-Center-Sigxxe
  table 14 name FLAC3D -Center-Sigxx
  table 15 name Error-Center-Sigxx
 end_command
 Num = 0
 zpnt = zone_head
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   if gp_xpos(gpnt) =0 then
    if gp_ypos(gpnt) = 0 then
    ; if z_xcen(zpnt) > 0 then
     ; if z_ycen(zpnt) < 0 then           ; IF No symmetry;
        mark = 1
     ; end_if
    ; end_if
    end_if
   end_if
  end_loop
  if mark = 1 THEN
   Num = Num +1
   global Depth =abs(z_zcen(zpnt))
   global m1 = abs(XLength/YLength)
   global n1 = abs(Depth/YLength)
   local A = sqrt(m1^2+n1^2+1)
   local B = m1^2+2*n1^2+1
   local C = m1^2+n1^2
   local D = 1+n1^2
   global I=(2/pi)*((m1*n1/A)*(B/(C*D))+ASIN(m1/(sqrt(C)*sqrt(D))))
   local R1=sqrt(XLength^2+Depth^2)
   local R2=sqrt(YLength^2+Depth^2)
   local R3=sqrt(XLength^2+YLength^2+Depth^2)
   local Iz5=(4/(2*pi))*(ATAN((-XLength*YLength)/(Depth*R3))+((-
   XLength*YLength*Depth/R3)*((1/R1^2)+(1/R2^2))))
   local Ix5=4*(1/(4*pi))*(atan((-XLength*YLength)/(Depth*R3))-((-2*XLength*YLength*Depth)/(R1^2*R3)))
```

```
    sigzze= -Iz5*Load
    sigxxe= -Ix5*Load
    sigzz = -z_szz(zpnt)
    sigxx = -z_sxx(zpnt)
    errz_center = ((sigzz - sigzze)/abs(sigzze))*100
    errx_center = ((sigxx - sigxxe)/abs(sigxxe))*100
    table(10,sigzze) =-Depth
    table(11,sigzz) = -Depth
    table(12,errz_center) = -Depth
    table(13,-Depth) =sigxxe
    table(14,-Depth) =sigxx
    table(15,errx_center) = -Depth
   end_if
   zpnt = z_next(zpnt)
 end_loop
 loop i(1,Num)
  local x1 = xtable(13,i)
  local y1 = ytable(13,i)
  local x2 = xtable(14,i)
  local y2 = ytable(14,i)
  xtable(13,i) = y1
  ytable(13,i) = x1
  xtable(14,i) = y2
  ytable(14,i) = x2
 end_loop
end
;-----------------------------------------------------------------------------------------
; exact and numerical solution for a uniform surface load in Semi-infinte medium
; Table 16: analytical values disze
; Table 17: numerical values disz
; at grid points under the corner of the load area
;-----------------------------------------------------------------------------------------
def nadis
 command
  table 16 name Analytic-Corner-uz
  table 17 name FLAC3D -Corner-uz
  table 18 name Error-Corner-uz
 end_command
 gpnt = gp_head
 loop while gpnt # null
  mark = 0
```

```
   if gp_xpos(gpnt) = XLength then
    if gp_ypos(gpnt) = YLength then
     if gp_zpos(gpnt) # 0 then
      mark = 1
     end_if
    end_if
   end_if
   if mark = 1 then
    global Depth =abs(gp_zpos(gpnt))
    local n1 = Depth/XLength1
    local m1 = XLength1/YLength1
    local s1 = sqrt(1+m1^2+n1^2)
    local A =(1/(2*pi))*(ln((s1+m1)/(s1-m1))+m1*ln((s1+1)/(s1-1)))
    local B =(n1/(2*pi))*atan(m1/(n1*s1))
    local nyp = (1-2*ny_Poisson)/(1-ny_Poisson)
    disze = ((Load*XLength1)/(E_Young))*(1- ny_Poisson^2)*(A-nyp*B);
    disz = gp_zdisp(gpnt)
    errdisz_corner = ((disz - disze)/abs(disze))*100
    table(16,disze) = -Depth
    table(17,disz) = -Depth
    table(18,errdisz_corner) = -Depth
   end_if
   gpnt = gp_next(gpnt)
 end_loop
 gpnt = gp_near(XLength,YLength,0)
 Bdissz = gp_zdisp(gpnt)
 local C = (4.0/pi)*ln(sqrt(2.0)+1.0)
 local D = Load*XLength*(1- ny_Poisson^2)
 Bdissze = (C*D)/E_Young
 Berrdiss =((Bdissz- Bdissze)/abs(Bdissze))*100
end
;-------------------------------------------------------------------------------------
; exact and numerical solution for a uniform surface load in Semi-infinte medium
; Table 19: analytical values disze
; Table 20: numerical values disz
; at grid points under the center of the load area
;-------------------------------------------------------------------------------------
def nadis1
 command
  table 19 name Analytic-Center-uz
  table 20 name FLAC3D -Center-uz
```

```
  table 21 name Error-Center-uz
 end_command
 gpnt = gp_head
 loop while gpnt # null
  mark = 0
  if gp_xpos(gpnt) = 0 then
   if gp_ypos(gpnt) = 0 then
    if gp_zpos(gpnt) # 0 then
     mark = 1
    end_if
   end_if
  end_if
  if mark = 1 then
   global Depth =abs(gp_zpos(gpnt))
   local n1 = Depth/XLength
   local m1 = XLength/(-YLength)
   local s1 = sqrt(1+m1^2+n1^2)
   local A =(1/(2*pi))*(ln((s1+m1)/(s1-m1))+m1*ln((s1+1)/(s1-1)))
   local B =(n1/(2*pi))*atan(m1/(n1*s1))
   local nyp = (1-2*ny_Poisson)/(1-ny_Poisson)
   disze = ((4*Load*XLength)/(E_Young))*(1- ny_Poisson^2)*(A-nyp*B);
   disz = gp_zdisp(gpnt)
   errdisz_center = ((disz - disze)/abs(disze))*100
   table(19,disze) = -Depth
   table(20,disz) = -Depth
   table(21,errdisz_center) = -Depth
  end_if
  gpnt = gp_next(gpnt)
 end_loop
 gpnt = gp_near(0,0,0)
 Adissz = gp_zdisp(gpnt)
 local C = (8.0/pi)*ln(sqrt(2.0)+1.0)
 local D = Load*XLength*(1- ny_Poisson^2)
 Adissze = (C*D)/E_Young
 Aerrdiss =((Adissz- Adissze)/abs(Adissze))*100
 ZoneNo=nzone
end
@nastr
@nastr1
@nadis
@nadis1
```

```
plot create view Stressz-Under-Load-Center
plot add table 10 , 11 ;
plot create view Stressx-Under-Load-Center
plot add table 13 , 14
plot create view Uz-Under-Load-Center
plot add table 19 , 20 ;
plot create view Stressz-Under-Load-Corner
plot add table 1 , 2
plot create view Stressx-Under-Load-Corner
plot add table 4 , 5
plot create view Stressxz-Under-Load-Corner
plot add table 7 , 8
plot create view Uz-Under-Load-Corner
plot add table 16 , 17
@test1
list @tim
pause
```

```
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Numerical solution for a circular tunnel in infinite elastic material
;                      (Theoretical solution by Lame)
;                       //Uncoupled Solution//
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
New
CONFIG cppfish
LOAD function BEMKMM002_64.dll
def testt
 tim0=clock
end
def test1
 tim=(clock-tim0)/100.0
end
@testt
def parm
 rad=1.0        ; radius of the tunnel
 d= 2.0
 len = d
 len2 = d/10
 in_size = 16    ; number of zones along outer edge
 in_size1 = 1    ; number of zones along the tunnel
 rad_size = 24   ; number of zones in radial direction
 xinitialstress = -1
 yinitialstress = 0
 zinitialstress = -1
 p =-xinitialstress
 radapp = len-0.01
 E_Young = 1000.0     ; Young Modulus
 ny_Poisson = 0.25    ; Poisson's Ratio
 Bulk_modulus = E_Young/(3*(1-2*ny_Poisson))
 G_kelvin = E_Young/(2*(1+ny_Poisson))
 global vx = 0.0
 global vy = 0.0
 global vz = 0.0
 radf = -rad
 lenf = -len
end
@parm
```

```
gen zone cshell size @in_size @in_size1 @rad_size 1 rat 1.0 1 1 1.0 p0 0 0 0 p1 @len 0 0 p2 0 @len2 0 &
p3 0 0 @len p4 @len @len2 0 p5 0 @len2 @len dim @rad @rad @rad @rad
;gen zone reflect dip 0 dd 90 origin 0 0 0  ;/if symmetry is about plane yz only/;

plot create view Tunnel
plot set name 'Tunnel'
plot add zone addlabel "Zone" yellow
plot boundary
plot add axes
model me elastic
prop bulk @Bulk_modulus shear @G_kelvin
ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress
apply sxx=@xinitialstress szz=@zinitialstress range cyl end1 0 0 0 end2 0 @len2 0 rad @len
apply remove sxx range cyl end1 0 0 0 end2 0 @len2 0 rad @rad
apply remove szz range cyl end1 0 0 0 end2 0 @len2 0 rad @rad
apply xv 0 range x -0.001 0.001 z @rad @len
;apply xv 0 range x -0.001 0.001 z @radf @lenf     ;/if symmetry is about plane yz only/;
apply zv 0 range z -0.001 0.001 x @rad @len
apply yv 0
hist n 5
hist add unbal
hist gp zdisp 0 0 @rad
hist gp xdisp @rad 0 0
def totalstepnumber1
 Whilestepping
 STEPNUMBER1 = STEPNUMBER1+1
end
solve
def totalstepnumber
 STEPNUMBER = STEPNUMBER1
end
@totalstepnumber
;------------------------------------------------------------------------
; exact and numerical solution for a Tunnel in infinte medium
; stores in
; Table 1: analytical radial stress sigre/p
; Table 3: numerical radial stress sigr/p
; Table 5: errsr the error in sigr
; Table 2: analytical tangential stress sigte/p
; Table 4: numerical tangential stress sigt/p
; at zone centroid closed to x axis
```

```
; Table 6: errst the error in sigt
; Table 7: analytical radial displacement ur at grid points
; Table 8: numerical radial displacement ur at grid points
; Table 9: err the error in ur
;-------------------------------------------------------------------------
def nastr
 command
  tab 1 name analytic-sigr
  tab 2 name analytic-sigt
  tab 3 name FLAC3D -sigr
  tab 4 name FLAC3D -sigt
  tab 5 name Error-in-sigr
  tab 6 name Error-in-sigt
 end_command
 radd = rad
 errsr = 0.0
 errst = 0.0
 zpnt = zone_head
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   if gp_zpos(gpnt) < 0.001 then
    ; if gp_zpos(gpnt) > -0.001 then        ;/if symmetry is about plane yz only/;
     mark = 1
    ; end_if                                ;/if symmetry is about plane yz only/;
   end_if
  end_loop
  radd = sqrt(z_xcen(zpnt)^2 + z_zcen(zpnt)^2)
  sigre = (1.-(rad/radd)^2)
  sigte = (1.+(rad/radd)^2)
  aux1 = (z_sxx(zpnt) + z_szz(zpnt)) * 0.5
  aux2 = sqrt(z_sxz(zpnt)^2 + 0.25 *(z_sxx(zpnt)-z_szz(zpnt))^2)
  sigr = -(aux1 + aux2) / p
  sigt = -(aux1 - aux2) / p
  errsr = ((sigr - sigre)/sigre)*100
  errst = ((sigt - sigte)/sigte)*100
  if mark = 1
   table(1,radd) = sigre
   table(2,radd) = sigte
   table(3,radd) = sigr
```

```
    table(4,radd) = sigt
    table(5,radd) = errsr
    table(6,radd) = errst
   end_if
   zpnt = z_next(zpnt)
  end_loop
end

def nadis
 command
  tab 7 name analytic-ur
  tab 8 name FLAC3D -ur
  tab 9 name Error-in-ur
 end_command
 radd = rad
 errd = 0.0
 pnt = gp_head
 loop while pnt # null
  mark = 0
  if gp_zpos(pnt) < 0.001 then
  ;if gp_zpos(pnt) > -0.001 then          ;/if symmetry is about plane yz only/;
   if gp_ypos(pnt) < 0.001 then
    mark = 1
   end_if
  ;end_if                                ;/if symmetry is about plane yz only/;
  end_if
  radd = sqrt(gp_xpos(pnt)^2 + gp_zpos(pnt)^2)
  dise =0.5*(p/G_kelvin)*(rad^2/radd)
  dis = sqrt(gp_xdisp(pnt)^2 + gp_zdisp(pnt)^2)
  err = ((dis-dise)/dise)*100
  if mark = 1 then
   table(7,radd) = dise
   table(8,radd) = dis
   table(9,radd) = err
  end_if
  pnt = gp_next(pnt)
 end_loop
 pnt = gp_near(0,0,rad)
 zdiss = gp_zdisp(pnt)
 ZoneNo=nzone
end
```

```
@nastr
@nadis
@test1
list @tim
plot create view stresses
plot set name 'Theoretical and Numerical Radial and Hoop Stresses'
plot add table 1 ,3,2,4
plot create view displacements
plot set name 'Theoretical and Numerical Radial Displacements'
plot add table 7,8
list @zdiss
list @ZoneNo
pause

;-------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------
; Numerical solution for a Hole in a plate in Elastic material {First configuration}
;                           (Theoretical solution by Mindlin)
;                               //Uncoupled Solution//
;-------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------
New
CONFIG cppfish
LOAD function BEMKMM002_64.dll
def testt
 tim0=clock
end
def test1
 tim=(clock-tim0)/100.0
end
@testt
def parm
 rad=5           ; radius of the hole (d/r= 10, elfa1= 2.0)
 d=50
 d1=-0.7*d
 len1=d+d1
 len2=-1         ; Cylinder outer radius
 len3=-d
 in_size= 16     ; number of zones along outer edge
 in_size1=1      ;number of zones along the tunnel
 rad_size= 24    ; number of zones in radial direction
```

```
 xinitialstress = 1.0
 yinitialstress = 0.0
 zinitialstress = 0.0
 radi= rad - 0.1
 radf= rad + 0.1
 E_Young = 1000.0    ; Young Modulus
 ny_Poisson = 0.3    ; Poisson's Ratio
 Bulk_modulus = E_Young/(3*(1-2*ny_Poisson))
 G_kelvin = E_Young/(2*(1+ny_Poisson))
 global vx = 0.0
 global vy = 0.0
 global vz = 0.0
 histP =len3+rad
end
@parm
gen zone cshell size @in_size @in_size1 @rad_size 1 rat 1.1 1.0 1.0 1 p0 0 @len2 @len3 p1 @len1 @len2
@len3 p2 0 0 @len3 & p3 0 @len2 @d1 p4 @len1 0 @len3 p5 0 0 @d1 dim @rad @rad @rad @rad
gen zone reflect dip 0 dd 90 origin 0 0 @len3
plot create view Hole
plot set name 'A hole in a plate'
plot add zone addlabel "Zone" yellow
plot boundary
plot add axes
model me elastic
prop bulk @Bulk_modulus shear @G_kelvin
ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress
apply sxx=@xinitialstress range cyl end1 0 0 @len3 end2 0 @len2 @len3 rad @len1
apply remove sxx range cyl end1 0 0 @len3 end2 0 @len2 @len3 rad @rad
apply xv 0 range x -0.001 0.001
hist n 5
hist add unbal
hist add gp xdisp @rad @len2 @len3
hist add gp zdisp 0 @len2 @histP
def totalstepnumber1
 Whilestepping
 STEPNUMBER1 = STEPNUMBER1+1
end
solve
def totalstepnumber
 STEPNUMBER = STEPNUMBER1
end
```

```
@totalstepnumber
list @stepnumber
@test1
list @tim
pause

;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------
; Numerical solution for a Hole in a plate in Elastic material{Second Configuration}
;                          (Theoretical solution by Mindlin)
;                               //Uncoupled Solution//
;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------
New
def testt
tim0=clock
end
def test1
tim=(clock-tim0)/100.0
end
@testt
def parm
 rad=0.5          ; radius of the hole (d/r= 10.0, elfa1= 2.0)
 d= 5.0
 d1=-0.0*d
 len1=d+d1
 len2=-(d+d1)/6   ; length of outer edge
 len3=-(d)
 len4=2*len3-d1
 len5=2.5*len4
 len6=-2*len4
 in_size= 72      ; number of zones along outer cube edge
 in_size1= 1      ; number of zones along the tunnel
 rad_size= 30     ; number of zones in radial direction
 rad_size2=rad_size/2
 xinitialstress=1.0
 yinitialstress=0.0
 zinitialstress=0.0
 xapp1=len6-0.001
 xapp2=len6+0.001
 zfix1=len5-0.001
```

```
 zfix2=len5+0.001
 E_Young=1000.0     ; Young Modulus
 ny_Poisson = 0.0   ; Poisson's Ratio
 Bulk_modulus=E_Young/(3*(1-2*ny_Poisson))
 G_kelvin=E_Young/(2*(1+ny_Poisson))
 histP =len3+rad
end
@parm
gen zone radcyl size 1 @in_size1 @rad_size @in_size rat 1 1 1 1.025 p0 0 @len2 @len3 p1 @len1 @len2 @len3
p2 0 0 @len3 &
p3 0 @len2 @d1 p4 @len1 0 @len3 p5 0 0 @d1 p6 @len1 @len2 @d1 p7 @len1 0 @d1 dim @rad @rad @rad @rad
gen zone reflect dip 0 dd 90 origin 0 0 @len3
gen zone radtunnel size @rad_size2 @in_size1 @rad_size @in_size rat 1 1 1 1.01 p0 0 0 @d1 p1 @len6 0 @d1
p2 0 @len2 @d1 &
p3 0 0 @len5 p4 @len6 @len2 @d1 p5 0 @len2 @len5 p6 @len6 0 @len5 p7 @len6 @len2 @len5 p8 @len1 0 @d1 &
p9 0 0 @len4 p10 @len1 @len2 @d1 p11 0 @len2 @len4 p12 @len1 0 @len4 p13 @len1 @len2 @len4
plot create view Hole
plot set name 'Hole'
plot add zone addlabel "Zone" yellow
plot boundary
plot add axes
plot show
model me elastic
prop bulk @Bulk_modulus shear @G_kelvin
ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress
apply sxx=@xinitialstress range x @xapp1,@xapp2
fix z range z @zfix1 @zfix2
fix x range x -.001 .001
hist n 5
hist add unbal
hist add gp xdisp @rad @len2 @len3
hist add gp zdisp 0 @len2 @histP
plot history 1
def totalstepnumber1
 Whilestepping
 STEPNUMBER1=STEPNUMBER1+1
end
solve
def totalstepnumber
 STEPNUMBER=STEPNUMBER1
end
```

```
@totalstepnumber
list @stepnumber
;-------------------------------------------------------------------------
; BEM and Flac3D  solution for a Plate in Semi-infinte Plate
;
; stores in
; Table 1: BEM values
; Table 2: Flac3D  values sigxx
; at zone centroid closed to Plate straight edge
;-------------------------------------------------------------------------
def nastr
 table(1, 0.00) = 9.79060000e-01
 table(1, 7.83072096e-01) = 9.80960000e-01
 table(1, 1.57590528e+00) = 9.86340000e-01
 table(1, 2.38875349e+00) = 9.94310000e-01
 table(1, 3.23291016e+00) = 1.00357000e+00
 table(1, 4.12137291e+00) = 1.01265000e+00
 table(1, 5.06971421e+00) = 1.02015000e+00
 table(1, 6.09729086e+00) = 1.02504000e+00
 table(1, 7.22900688e+00) = 1.02677000e+00
 table(1, 8.49799552e+00) = 1.02538000e+00
 table(1, 9.94987437e+00) = 1.02143000e+00
 table(1, 1.16498061e+01) = 1.01587000e+00
 table(1, 1.36948272e+01) = 1.00983000e+00
 table(1, 1.62367193e+01) = 1.00435000e+00
 table(1, 1.95277280e+01) = 1.00023000e+00
 table(1, 2.40211217e+01) = 9.97830000e-01
 table(1, 3.06225645e+01) = 9.97090000e-01
 table(1, 4.14442094e+01) = 9.97580000e-01
 table(1, 6.28210344e+01) = 9.98650000e-01
 command
  table 1 name BEM-Sigxx
  table 2 name FLAC3D -Sigxx
 end_command
 zpnt = zone_head
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   if gp_ypos(gpnt) >= -0.001 then
    if gp_zpos(gpnt) >= -0.001 then
```

```
      mark =mark+ 1
     end_if
    end_if
   end_loop
   if mark = 2 THEN
    XR= z_xcen(zpnt)/rad
    sigxx = z_sxx(zpnt)/xinitialstress
    table(2,XR) = sigxx
   end_if
   zpnt = z_next(zpnt)
 end_loop
end
@nastr
@test1
list @tim
plot create view Stress-At-Straight-Edge
plot add table 1 , 2
pause
```

```
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Numerical solution for a smooth square footing on Tresca material problem
;                       -associated plastic flow-
;                        //Uncoupled Solution//
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
  New
  CONFIG cppfish
  LOAD function BEMKMM002_64.dll
  def testt
   global tim0=clock
  end
  def test1
   global tim=(clock-tim0)/100.0
  end
  @testt
  def parm
   global len = 9.0   ; length of outer box edge
   global in_size = 8 ; number of zones along outer cube edge
   global STEPNUMBER = 8000
   global d_a = 3.0
   global d_b = 3.0
   global rad = d_b
   global rab = d_b/d_a
   global eps = 0.1
   global ae = d_a + eps
   global be = d_b + eps
  end
  @parm
  gen zone radbrick size @in_size @in_size @in_size @in_size rat 1.0 1.0 1.0 1 p0 0 0 0 p1 @len 0 0 p2 0
  @len 0 &
  p3 0 0 @len p4 @len @len 0 p5 0 @len @len p6 @len 0 @len p7 @len @len @len dim @rad @rad @rad
  def make_sphere
  ; Loop over all GPs and remap their coordinates:
  ; assume len>rad
   p_gp=gp_head
   loop while p_gp#null
   ; Get gp coordinate: P=(px,py,pz)
    px=gp_xpos(p_gp)
    py=gp_ypos(p_gp)
```

```
    pz=gp_zpos(p_gp)
    maxp=max(abs(px),max(abs(py),abs(pz)))
    ; Compute A=(ax,ay,az)=point on sphere radially "below" P.
    dist=sqrt(px*px+py*py+pz*pz)
    if dist> 0 then
     k=rad/maxp
     ax=px*k
     ay=py*k
     az=pz*k
     ; Compute B=(bx,by,bz)=point on outer box boundary radially "above" P.
     maxp=max(abs(px),max(abs(py),abs(pz)))
     ;k=len/maxp
     k=len/dist
     bx=px*k
     by=py*k
     bz=pz*k
     ; Linear interpolation: P=A+u*(B-A)
     u=(maxp-rad)/(len-rad)
     gp_xpos(p_gp)=ax+u*(bx-ax)
     gp_ypos(p_gp)=ay+u*(by-ay)
     gp_zpos(p_gp)=az+u*(bz-az)
    end_if
    p_gp=gp_next(p_gp)
  end_loop
end
@make_sphere
gen zone brick size @in_size @in_size @in_size rat 1.0 1.0 1.0 p0 0 0 0 p1 @rad 0 0 p2 0 @rad 0 p3 0 0
@rad ;gen zone reflect dip 90 dd 90 origin 0 0 0
;gen zone reflect dip 90 dd 180 origin 0 0 0
plot create view Brick
plot set name 'Brick'
plot add zone addlabel "Zone" yellow
plot boundary
plot add axes
def fixboundary
 p_gp =gp_head
 loop while p_gp # null
  distance = sqrt(gp_xpos(p_gp)^2+gp_ypos(p_gp)^2+gp_zpos(p_gp)^2)
  dv=gp_id(p_gp)
  if distance >= len-0.001 then
   COMMAND
```

```
      Apply xv 0 range id @dv
      Apply yv 0 range id @dv
      Apply zv 0 range id @dv
     END_COMMAND
    end_if
    p_gp=gp_next(p_gp)
  end_loop
end
@fixboundary
model mech mohr
prop bul 2.e8 shea 1.e8 cohesion 1.e5
prop friction 0. dilation 0. tension 1.e10
fix z range x -0.1 @ae y -0.1 @be z -0.1 0.1
ini zvel 2.5e-5 range x -0.1 @ae y -0.1 @be z -0.1 0.1
apply xv 0 range x -.001 .001
apply yv 0 range y -.001 .001
;-----------------------------------------------------------------------
; p_load : average footing pressure / c
; p_solup : upper bound value for the bearing capacity / c
; p_sollo : lower bound value for the bearing capacity / c
; c_disp : vertical displacement at footing center / a
;-----------------------------------------------------------------------
def p_cons
 if rab > 0.53 then
  global p_solup = 5.24 + 0.47 * rab
 else
  p_solup = 5.14 + 0.66 * rab
 end_if
 global p_sollo = 2.00 + pi
 local d_a1 = d_a + 1.
 local d_b1 = d_b + 1.
 global pdis0 = gp_near(0.0,0.0,0.0)
 local pdis1 = gp_near(d_a1,0.0,0.0)
 local pdis2 = gp_near(0.0,d_b1,0.0)
 global area = (gp_xpos(pdis1)+d_a)*(gp_ypos(pdis2)+d_b)*0.25
end
@p_cons
def p_load
 local pnt = gp_head
 local pload = 0.0
 local n = 0
```

```
  loop while pnt # null
   if gp_zpos(pnt) < eps then
    if gp_xpos(pnt) < ae then
     if gp_ypos(pnt) < be then
      pload = pload + gp_zfunbal(pnt)
      n = n + 1
     end_if
    end_if
   end_if
   pnt = gp_next(pnt)
  end_loop
  pload = - pload / (area * z_prop(zone_head,'cohesion'))
  p_load = pload
  global c_disp = gp_zdisp(pdis0) / d_a
  global p_errlo = 100. * (pload - p_sollo) / p_sollo
  NoZones =nzone
end
hist nstep 50
hist add fish @p_load
hist add fish @p_solup
hist add fish @p_sollo
hist add fish @c_disp
hist add unbal
cyc @STEPNUMBER
save sslab
plot create view stress_displacement
plot add hist 1 2 3 vs 4
def const
 local bm = z_prop(zone_head,'bulk')
 local sm = z_prop(zone_head,'shear')
 global ny_Poisson = (3.*bm-2.*sm)/(6.*bm+2.*sm)
 global E_Young=3*bm*(1-2*ny_Poisson)
end
@const
plot create view shear_state
plot add zone colorby state
plot create view DispCont
plot set dip 90 dd 90 center 0 0 0
plot add cont disp plane behind
plot add axes
plot create view stresscontours
```

```
plot set dip 90 dd 90 center 0 0 0
plot add zonecont szz
plot add axes
def safetyfactor
;-------------------------------------------------------------------------------
; Obtains the inverse of Factor of Saftey [fstr] inside Flac3D Sub-domain
;
; str1: minor principal stress
; str3: major principal stress
; PHI: friction angle
; coh: cohesion
;-------------------------------------------------------------------------------
 command
  tab 1 name Fs_in_X_direction
 end_command
 pnt = zone_head
 loop while pnt # null
  loop igp (1,8)
   gpnt = z_gp(pnt,igp)
   mark = 0
   if gp_ypos(gpnt) = 0.0 then
    if gp_zpos(gpnt) = 0.0 then
     mark = mark+1
    end_if
   end_if
   if mark = 1 THEN
    str1 = (z_sig1(pnt))
    str3 = (z_sig3(pnt))
    PHI = z_prop(pnt,'friction')
    coh = z_prop(pnt,'cohesion')
    NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
    fstr=(str3 - str1)/(2*coh*sqrt(NPHI)+(str3)*(NPHI-1))
    radd= z_xcen(pnt)
    table(1,radd) = fstr
   end_if
  end_loop
  pnt = z_next(pnt)
 end_loop
 command
  tab 2 name Fs_in_Z_direction
 end_command
```

```
pnt = zone_head
loop while pnt # null
 loop igp (1,8)
  gpnt = z_gp(pnt,igp)
  mark = 0
  if gp_xpos(gpnt) = 0.0 then
   if gp_ypos(gpnt) = 0.0 then
    mark = mark+1
   end_if
  end_if
  if mark = 1 THEN
   str1 = (z_sig1(pnt))
   str3 = (z_sig3(pnt))
   PHI = z_prop(pnt,'friction')
   coh = z_prop(pnt,'cohesion')
   NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
   fstr=(str3 - str1)/(2*coh*sqrt(NPHI)+(str3)*(NPHI-1))
   radd= z_zcen(pnt)
   table(2,radd) = fstr
  end_if
 end_loop
 pnt = z_next(pnt)
end_loop
command
 tab 3 name Fs_in_Y_direction
end_command
pnt = zone_head
loop while pnt # null
 loop igp (1,8)
  gpnt = z_gp(pnt,igp)
  mark = 0
  if gp_xpos(gpnt) = 0.0 then
   if gp_zpos(gpnt) = 0.0 then
    mark = mark+1
   end_if
  end_if
  if mark = 1 then
   str1 = (z_sig1(pnt))
   str3 = (z_sig3(pnt))
   PHI = z_prop(pnt,'friction')
   coh = z_prop(pnt,'cohesion')
```

```
      NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
      fstr=(str3 - str1)/(2*coh*sqrt(NPHI)+(str3)*(NPHI-1))
      radd= z_ycen(pnt)
      table(3,radd) = fstr
    end_if
   end_loop
   pnt = z_next(pnt)
  end_loop
end
@safetyfactor
@test1
list @tim
plot create view Fs-in-X-direction
plot add table 1
plot create view Fs-in-Z-direction
plot add table 2
plot create view Fs-in-Y-direction
plot add table 3
list tab 1
list tab 2
list tab 3
list @p_sollo @p_load @p_solup
list @p_errlo
list @NoZones
pause
```

Variables, Parameters and Arrays Used in the Following Coupled Solution Codes:

NELM is the total number of interface boundary elements.
CD is the problem dimension (2D or 3D).
ELD is the boundary element dimension (1D or 2D).
DOF is the degree of freedom of a boundary node.
DOFL is the total number of D.O.Fs for a boundary element.
TNODDF is the total D.O.Fs of the interface.
SPS is the analysis code for 2D problems:
Solid plane strain problems = 1, solid plane stress problems = 2.
TYSUB is the type of BEM Sub-domain code:
Finite = 1, Infinite = 2, Semi-infinite = 3.
KelMind is the method of analysis code for the semi-infinite BEM Sub-domain or the type of fundamental solutions used:
Kelvin's = 0, Mindlin's = 1, Melan's = 2.
TYM is the Symmetry code of the planes:
yz = 1, yz and xz = 2, yz, xz and xy =3.
ELTY is the boundary element type = 1 for linear BEs.
NDEL is the number of nodes of a boundary element.
E_Young is Young modulus.
ny_Poisson is Poisson's ratio.
iterations is the maximum number of iterations.
relaxation is the relaxation parameter.
Factorepsolon is the correction ratio of the solution's initial error.
VCD is the total number of the stress components at a boundary node.
INTP is the total number of internal points, where the displacement and stress are requested to be computed, in the unbounded BEM sub-domain (post-processing).
NELI, NELF are the first and last boundary element number, respectively; the stress is requested to be commuted at as a post-processing step.
TNOD is the nodes total number of interface.
TNODTempf is the maximum boundary nodes number. This number is obtained from the Flac^{3D} program.
SHFUN(NDEL) is a boundary element shape function vector of dimension; it equals to NDEL.
DESHFUN(NDEL,ELD) is a BE derivatives array of its shape function vector.
Coord (CD) is a boundary node Cartesian coordinate's vector.
CoordL(CD,NDEL) is an array of a BE nodes' Cartesian coordinates.
v_Normalize(CD) is a normalized vector.
xproduct1(CD) is an x product vector of vectors **v1_xproduct** and **v2_xproduct**.
v_Outnormal(CD) is a vector normal to a point on a BE in xsai and etta directions (intrinsic coordinates).
v1_Outnormal(CD) and v2_Outnormal(CD) are Vectors in xsai and etta directions on a boundary element.
v_BIC(CD) and **Coord_BIC(CD)** are a local normal and coordinate vectors, respectively used by a Fish function called BoundaryIncidences.

The following arrays and vectors are either computed or prepared to be used by the BEM code, which performs the numerical analysis over the unbounded BEM sub-domain:

Incd(NELM,NDEL) is the interface boundary elements incidences array.
BELU(NELM,DOFL) and **BELT(NELM,DOFL)** are the interface boundary elements displacement and traction arrays, respectively.
BELU_Mindlin(NELM,DOFL) is the interface boundary elements displacement array in the BEM Cartesian coordinate system of Mindlin's and Melan's fundamental solutions.
NCCrd(CD,TNOD) is the Boundary (the interface) Node coordinates array.
NCCrd_Mindlin(CD,TNOD) is the Boundary Node coordinates array in Mindlin's and Melan's BEM Cartisian system.
NCCrd_INT(INTP,CD) is the internal Nodes coordinates array.
StressBE(NELM,NDEL,VCD) is the boundary elements stress components array.
U_INT(INTP,CD) is the array of displacement vectors at chosen internal nodes.
Stress_INT(INTP,VCD) is the stress components array at chosen internal nodes.

The following arrays and vectors are either computed or prepared to be used by the Flac^{3D} program, which performs the numerical analysis over the bounded FDM sub-domain:

BEL(NELM) is the vector of BEs Flac^{3D} numbers(on the interface).
IncdTemp(NELM,NDEL) is a temporary array of the interface BEs incidences as numerated by the Flac^{3D} program.
IncdTemp1(NELM,NDEL) is another temporary array of the interface BEs incidences. It is only used in the 2D problems (this array is extracted from Flac^{3D} for the other side of the 3D interface, which is divided into two one-dimension boundaries).
IncdFlac(NELM,NDEL1) and **IncdFlacT(NELM,NDEL1)** are temporary arrays of the inteface BEs incidences to be transformed into a BEM incidence array later in the code.
IncdFlac1(NELM,NDEL1) and **IncdFlacT1(NELM,NDEL1)** are temporary arrays of the interface BEs incidences used only in 2D problems.
NCCrd_Flac(CD,TNODTempf) is a temporary array of the Boundary Nodes Flac^{3D} coordinates.
NCCrd_Flac1(CD,TNODTempf1) is another temporary array of the Boundary Nodes Flac^{3D} coordinates used only in 2D problems.
IncdLFlac(NDEL) is an array of a BE nodes' Flac^{3D} coordinates.
BELU_Flac(NELM,DOFL,iterations1) is the interface boundary elements displacement array (from the Flac^{3D} side) at all iterations.
BELf_Flac (TNODDOF) is the interface nodal forces vector. This vector is extracted from the Flac^{3D} program.
BELf_FlacMindlin(TNODDOF) is the above vector modified to be suitable for Minlin's and Melan's BEM coordinate system.
epselontolEL(iterations1) is the error vector at all iterations.

```
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
; Numerical solution for a spherical Excavation in Elastic material
;     (Theoretical solution by R. V. Southwell (1926) problem
;                        //Coupled Solution//
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------

def Element_nunmber
;---------------------------------------------------------------
;  Obtain The Boundary Elements Number NELM [1]
;---------------------------------------------------------------
 zpnt = zone_head
 NELM=0
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+gp_ypos(gpnt)^2+gp_zpos(gpnt)^2)
   if distance >= len-0.001 then
    mark = mark+1
   end_if
  end_loop
  if mark = 4 THEN
   NELM = NELM+1
  end_if
  zpnt = z_next(zpnt)
 end_loop
end
@Element_nunmber

def main_input
;-------------------------------------------------------
;  Assign the basic information[2]
;-------------------------------------------------------
 array ar(1) arr(1) arrr(3,4) arrrr(1)
 filename1 = 'result.dat'
 oo = open(filename1,1,1)
 arr(1)='Project: Spherical Excavation'
 oo=write(arr,1)
 CD = 3           ; Cartesian dimension (2D & 3D )
```

```
arr(1)='Cartesian_dimension ='+string(CD)
oo=write(arr,1)
ELD= CD-1          ; Boundary Element dimension
DOF = 3            ; Degrees of freedom
TYSUB = 2          ; Type of Sub-domain (Finite BEM Sub-domain = 1, Infinite BEM Sub-domain = 2,Semi-
Infinite BEM Sub-domain = 3)
KelMind= 0         ; Method of Analysis for Semi-infinte Sub-domain or Type of Used Fundamental solutions(
Kelvin's = 0, Mindlin's = 1, Melan's = 2)
IF TYSUB = 1 THEN
 arr(1)='Finite Region'
 oo=write(arr,1)
ELSE
 IF TYSUB = 2 THEN
  arr(1)='Infinite Region'
  oo=write(arr,1)
 ELSE
  arr(1)='Semi-Infinite Region'
  oo=write(arr,1)
  IF KelMind=0
   arr(1)='Semi-Infinite Region/ Kelvin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=1
   arr(1)='Semi-Infinite Region/Mindlin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=2
   arr(1)='Semi-Infinite Region/Melan solution '
   oo=write(arr,1)
  END_IF
 END_IF
END_IF
TYM = 2              ; Symmetry about yz = 1, Symmetry about yz and xz = 2, Symmetry about yz, xz and xy
=3)
CASEOF TYM
 arr(1)='No symmetry'
 oo=write(arr,1)
 CASE 1
  arr(1)='Symmetry about y-z plane'
  oo=write(arr,1)
 CASE 2
```

```
   arr(1)='Symmetry about y-z and x-z planes'
   oo=write(arr,1)
  CASE 3
   arr(1)='Symmetry about all planes'
   oo=write(arr,1)
ENDCASE
SPS= 1
ELTY=1              ; Element type
; Determine number of nodes per element NDEL
IF CD = 2 THEN
 NDEL= 2
ELSE
 NDEL= 4
END_IF
arr(1)=' Single Element Nodes number = '+ string(NDEL)
oo=write(arr,1)
; Material Properties
arr(1)=' Young modulus = '+string(E_Young)
oo=write(arr,1)
arr(1)=' Poisson ratio= '+string(ny_Poisson)
oo=write(arr,1)
iterations = 50     ; No. of iterations
iterations1=iterations+1  ; Temp
arr(1)=' Number of Iterations= '+string(iterations)
oo=write(arr,1)
relaxation=0.4
arr(1)=' relaxation Parameter= '+string(relaxation)
oo=write(arr,1)
iteration_Tolerance = 0.00005 ;iteration tolerance should be higher if function NextiterFlacvelocity1 is
used (0.0002)
arr(1)=' iteration Tolerance= '+string(iteration_Tolerance)
oo=write(arr,1)
Factorepsolon = 0.01
VCD= 2*CD                   ; No. of Stress Components at a Node
DOFL= NDEL*DOF              ; Total degrees of freedom of element
INTP = 29                   ; Number of internal points displacement and stress are required to be
computed at
NELI= 1
NELF= NELM
NDEL1= NDEL+1
oo = close
```

```
end
@main_input
def Matrices_dimensions1
;-------------------------------------------------------------------------
; Define the Program Arrays Dimensions
;-------------------------------------------------------------------------
 array SHFUN(NDEL) DESHFUN(NDEL,ELD)   ; Shape function & Derivatives of shape function
 array CoordL(CD,NDEL) Coord(CD)       ; element coordinates & Cartesian coordinates
 array v_Normalize(CD) v1_xproduct(CD) v2_xproduct(CD) xproduct1(CD)
 array v_Outnormal(CD) v1_Outnormal(CD) v2_Outnormal(CD  ;Vector normal to point & Vectors in xsai,etta
 directions
 array Incd(NELM,NDEL) v_BIC(CD) Coord_BIC(CD)
 array BELU(NELM,DOFL) BELT(NELM,DOFL)
 array BELU_Flac(NELM,DOFL,iterations1) BELU_Mindlin(NELM,DOFL)
end
@Matrices_dimensions1
def Temporaryarray
;-------------------------------------------------------------------------
; Obtain Boundary Elements Temporary Incidences Array[3]
;-------------------------------------------------------------------------
 Array BEL(NELM)
 Array IncdTemp(NELM,NDEL)
 zpnt = zone_head
 BEN=1
 loop while zpnt # null
  mark = 0
  N=0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+gp_ypos(gpnt)^2+gp_zpos(gpnt)^2)
   if BEN <= NELM Then
    if distance >= len-0.001 then
     N=N+1
     IncdTemp(BEN,N) = gp_id(gpnt)
     TNODTempf=Max(TNODTempf,IncdTemp(BEN,N))
     mark =mark+1
    end_if
   end_if
  end_loop
  if mark = 4 THEN
   BEL(BEN)=z_id(zpnt)
```

```
    NELMTemp=Max(NELMTemp,BEL(BEN))
    BEN = BEN + 1
   end_if
   zpnt = z_next(zpnt)
 end_loop
end
@Temporaryarray
def BoundaryArrayFlac
;---------------------------------------------------------------------
; Obtain Boundary Nodes Temporary Coordinates Array[4]
;---------------------------------------------------------------------
 Array IncdFlac(NELM,NDEL1)
 Array IncdFlacT(NELM,NDEL1)
 Array NCCrd_Flac(CD,TNODTempf)
 Array IncdLFlac(NDEL)
 loop BEN(1,NELM)
  IncdFlac(BEN,1)=BEL(BEN)
  loop m(2,NDEL1)
   IncdFlac(BEN,m)=IncdTemp(BEN,m-1)
   IncdFlacT(BEN,m)=IncdFlac(BEN,m)
  end_loop
  Temporary1=IncdFlac(BEN,4)
  Temporary2=IncdFlac(BEN,5)
  IncdFlac(BEN,4)=Temporary2
  IncdFlac(BEN,5)=Temporary1
  IncdFlacT(BEN,4)=Temporary2
  IncdFlacT(BEN,5)=Temporary1
 end_loop
 loop m(1,NELM)
  Loop n(2,NDEL1)
   IF IncdFlac(m,n)#0 THEN
    gpnt=find_gp(IncdFlac(m,n))
    NCCrd_Flac(1,IncdFlac(m,n))=gp_xpos(gpnt)
    NCCrd_Flac(2,IncdFlac(m,n))=gp_ypos(gpnt)
    NCCrd_Flac(3,IncdFlac(m,n))=gp_zpos(gpnt)
   END_IF
  end_loop
 end_loop
end
@BoundaryArrayFlac
```

```
def Shape_Function
;---------------------------------------------------------------
; Obtain shape functions
;---------------------------------------------------------------
 loop n (1,NDEL)
  SHFUN(n)=0
 end_loop
 IF CD = 2 THEN
  SHFUN(1)= 0.5*(1.0 - xsai)
  SHFUN(2)= 0.5*(1.0 + xsai)
 ELSE
  SHFUN(1)= 0.25*(1.0-xsai)*(1.0-etta)
  SHFUN(2)= 0.25*(1.0+xsai)*(1.0-etta)
  SHFUN(3)= 0.25*(1.0+xsai)*(1.0+etta)
  SHFUN(4)= 0.25*(1.0-xsai)*(1.0+etta)
 END_IF
END
def Derive_Shape_Function
;---------------------------------------------------------------
; Obtain Derivatives of shape functions
;---------------------------------------------------------------
 loop m (1,ELD)
  loop n (1,NDEL)
   DESHFUN(n,m) =0
  end_loop
 end_loop
 IF CD=2 THEN
  DESHFUN(1,1)= -0.5
  DESHFUN(2,1)= 0.5
 ELSE
  DESHFUN(1,1)= -0.25*(1.0-etta)
  DESHFUN(1,2)= -0.25*(1.0-xsai)
  DESHFUN(2,1)= 0.25*(1.0-etta)
  DESHFUN(2,2)= -0.25*(1.0+xsai)
  DESHFUN(3,1)= 0.25*(1.0+etta)
  DESHFUN(3,2)= 0.25*(1.0+xsai)
  DESHFUN(4,1)= -0.25*(1.0+etta)
  DESHFUN(4,2)= 0.25*(1.0-xsai)
 END_IF
END
def Coordinate
```

```
;-----------------------------------------------------
; Obtain Cartesian coordinates
;-----------------------------------------------------
 LOOP n (1,CD)
  Coord(n)=0
 END_LOOP
 LOOP i (1,CD)
  LOOP j(1,NDEL)
   Coord(i)= Coord(i)+CoordL(i,j)*SHFUN(j)
  END_LOOP
 END_LOOP
END
def Normalize
;---------------------------------------
; Normalise vector
;---------------------------------------
 IF CD=2 THEN
  M_Normalize= SQRT(v_Normalize(1)^2+v_Normalize(2)^2)
 ELSE
  M_Normalize= SQRT(v_Normalize(1)^2+v_Normalize(2)^2+v_Normalize(3)^2)
 END_IF
 IF M_Normalize = 0 THEN
  EXIT
 END_IF
 LOOP n (1,CD)
  v_Normalize(n)= v_Normalize(n)/M_Normalize
 END_LOOP
END
def x_product
;---------------------------------------
; x-product v1xv2
;---------------------------------------
 xproduct1(1)=V1_xproduct(2)*V2_xproduct(3)-V2_xproduct(2)*V1_xproduct(3)
 xproduct1(2)=V1_xproduct(3)*V2_xproduct(1)-V1_xproduct(1)*V2_xproduct(3)
 xproduct1(3)=V1_xproduct(1)*V2_xproduct(2)-V1_xproduct(2)*V2_xproduct(1)
END
def Outnormal
;-----------------------------------------------------
; Obtains the outward normal vector
;-----------------------------------------------------
 LOOP n (1,CD)
```

```
  v1_Outnormal(n) =0
  v2_Outnormal(n) =0
 END_LOOP
 xsai=xsai_Outnormal
 etta=etta_Outnormal
 Derive_Shape_Function
 LOOP i (1,CD)
  LOOP j(1,NDEL)
   V1_Outnormal(i)= V1_Outnormal(i)+CoordL(i,j)*DESHFUN(j,1)
   IF CD=3 THEN
    V2_Outnormal(i)= V2_Outnormal(i)+CoordL(i,j)*DESHFUN(j,2)
   END_IF
  END_LOOP
 END_LOOP
 LOOP n(1,CD) ; normal vector
  V1_xproduct(n)= v1_Outnormal(n)
  V2_xproduct(n)= v2_Outnormal(n)
 end_loop
 IF CD = 2 THEN
  v_Outnormal(1)= V1_xproduct(2)
  v_Outnormal(2)= -V1_xproduct(1)
 ELSE
  x_product
  LOOP n(1,CD)
   v_Outnormal(n)= xproduct1(n)
  end_loop
 END_IF
 LOOP n(1,CD) ; Normalise
  v_Normalize(n)=v_Outnormal(n)
 END_LOOP
 Normalize
 LOOP n(1,CD)
  v_Outnormal(n) = v_Normalize(n)
 END_LOOP
END
def BoundaryIncidences
;---------------------------------------------------------------------
; Obtain the Boundary Elements Outward Normal and
; Their Incidences Array [5]
;---------------------------------------------------------------------
 LOOP BEN(1,NELM)
```

```
LOOP m(2,NDEL1)
 LOOP n (1,CD)
  CoordL(n,m-1)= NCCrd_Flac(n,IncdFlac(BEN,m))
 END_LOOP
 IncdLFlac(m-1)= IncdFlac(BEN,m)
END_LOOP
LOOP SC (1,CD)
 v_Outnormal(SC) =0
END_LOOP
xsai_Outnormal=0
etta_Outnormal=0
Outnormal
LOOP SC (1,CD)
 v_BIC(SC)= v_Outnormal(SC)
END_LOOP
Shape_Function
Coordinate
LOOP SC (1,CD)
 Coord_BIC(SC)= Coord(SC)
END_LOOP
Costh_BIC=0
loop Sc (1,CD)
 Costh_BIC=Costh_BIC+ V_BIC(Sc)*Coord_BIC(Sc)
end_loop
IF CD = 3 THEN
 Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2+Coord_BIC(3)^2)
ELSE
 Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2)
END_IF
Costh_BIC=Costh_BIC/Costh
IF CD = 3 THEN
 IF Costh_BIC > 0 Then
  IncdFlac(BEN,2)= IncdFlacT(BEN,5)
  IncdFlac(BEN,3)= IncdFlacT(BEN,4)
  IncdFlac(BEN,4)= IncdFlacT(BEN,3)
  IncdFlac(BEN,5)= IncdFlacT(BEN,2)
 END_IF
END_IF
IF CD = 2 THEN
 IF Costh_BIC > 0 Then
  IncdFlac(BEN,2)= IncdFlacT(BEN,3)
```

```
    IncdFlac(BEN,3)= IncdFlacT(BEN,2)
   END_IF
  END_IF
 END_LOOP
 LOOP BEN(1,NELM)
  LOOP m(1,NDEL1)
   IncdFlacT(BEN,m)= IncdFlac(BEN,m)
  End_LOOP
 End_LOOP
 I=0
 LOOP BEN(1,NELM)
  LOOP m(2,NDEL1)
   I=I+1
   IncdFlac(BEN,m)= I
   IF BEN >1 THEN
    SECTION
     LOOP s(1,BEN-1)
      LOOP mm(2,NDEL1)
       IF IncdFlacT(s,mm)=IncdFlacT(BEN,m) THEN
        IncdFlac(BEN,m)=IncdFlac(s,mm)
        I=I-1
        EXIT SECTION
       END_IF
      End_LOOP
     End_LOOP
    END_SECTION
   END_IF
  End_LOOP
 End_LOOP
 loop BEN(1,NELM)
  loop m(2,NDEL1)
   Incd(BEN,m-1)=IncdFlac(BEN,m)
   TNOD=Max(TNOD,Incd(BEN,m-1))
  end_loop
 end_loop
 TNODDOF=TNOD*DOF
 oo = open(filename1,2,1)
 arr(1)='Number of Nodes of The System= '+string(TNOD)
 oo=write(arr,1)
 arr(1)='Number of Elements of The System= '+string(NELM)
 oo=write(arr,1)
```

```
 oo=close
end
@BoundaryIncidences
def NodesCoordinates
;-------------------------------------------------------------------------
; Obtain Boundary Nodes Coordinates [6]
;-------------------------------------------------------------------------
 array NCCrd(CD,TNOD) ; Boundary Node co-ordinates
 array NCCrd_Mindlin(CD,TNOD)
 LOOP BEN(1,NELM)
  LOOP m(2,NDEL1)
   LOOP n (1,CD)
    NCCrd(n,Incd(BEN,m-1))= NCCrd_Flac(n,IncdFlacT(BEN,m))
    if abs(NCCrd(n,Incd(BEN,m-1))) < 10e-10 then
     NCCrd(n,Incd(BEN,m-1)) = 0.0
    end_if
    if NCCrd(1,Incd(BEN,m-1))= len
     if NCCrd(2,Incd(BEN,m-1))=0
      if NCCrd(3,Incd(BEN,m-1))=0
       BENiter = BEN
       miter = m-1
      end_if
     end_if
    end_if
   END_LOOP
  END_LOOP
 END_LOOP
 IF TYSUB = 3 THEN
 LOOP m(1,TNOD)
 IF CD=3 THEN
 NCCrd_Mindlin(2,m) = NCCrd(1,m)
 NCCrd_Mindlin(1,m) =-NCCrd(3,m)
 NCCrd_Mindlin(3,m) =-NCCrd(2,m)
 ELSE
 NCCrd_Mindlin(2,m) = NCCrd(1,m)
 NCCrd_Mindlin(1,m) =-NCCrd(2,m)
 END_IF
 END_LOOP
 END_IF
end
@NodesCoordinates
```

```
def Matrices_dimensions2
;-------------------------------------------------------------------------
; Define the Program Matrices Dimensions
;-------------------------------------------------------------------------
 array BELf_Flac(TNODDOF)BELf_FlacMindlin(TNODDOF)
 array NCCrd_INT(INTP,CD) StressBE(NELM,NDEL,VCD)
 array U_INT(INTP,CD) Stress_INT(INTP,VCD)
 array epselontolEL(iterations1)
end
@Matrices_dimensions2
def Input_INT
;-------------------------------------------------------------------------
; Reads Internal Points Coordinates and Writes Boundary Nodes Coordinates
;-------------------------------------------------------------------------
 filename2 = 'Input.dat'
 oo = open(filename2,0,1)
 loop n (1,INTP)
  loop m (1,CD)
   oo=read(ar,1)
   NCCrd_INT(n,m)=float(ar(1))
  endloop
 endloop
 oo=close
 oo = open(filename1,2,1)
 arr(1)='Nodes Coordinates:'
 oo=write(arr,1)
 loop m (1,TNOD)
  arr(1)=' '
  loop n (1,CD)
   IF TYSUB # 3 THEN
    arr(1) =string(arr(1))+' '+ string(NCCrd(n,m))
   ELSE
    arr(1) =string(arr(1))+' '+ string(NCCrd_Mindlin(n,m))
   ENDIF
  end_loop
  ar(1)='Node No.'+string(m)
  oo = write(ar,1)
  oo = write(arr,1)
 end_loop
 arr(1)='Elements Incidences:'
 oo=write(arr,1)
```

```
 loop m (1,NELM)
  arr(1)=' '
  loop n (1,NDEL)
   arr(1) =string(arr(1))+' '+string(Incd(m,n))
  end_loop
  ar(1)='ELEMENT No.'+string(m)
  oo = write(ar,1)
  oo = write(arr,1)
 end_loop
 arr(1)='Coordinates of points in unbounded domain:'
 oo=write(arr,1)
 loop n(1,INTP)
  arr(1)=' '
  loop m(1,CD)
   arr(1) =string(arr(1))+' '+string(NCCrd_INT(n,m))
  end_loop
  ar(1)='point No.'+string(n)
  oo = write(ar,1)
  oo = write(arr,1)
 end_loop
 oo=close
END
def writing1
;---------------------------------------
; Write Some of FISH Functions
;---------------------------------------
 oo = open(filename1,2,1)
 arr(1)=' '
 oo = write(arr,1)
 loop BEN (1,NELM)
  ar(1)='RESULTS OF ELEMENT No.'+string(BEN)
  oo = write(ar,1)
  arr(1)= 'Displacement u:'
  oo=write(arr,1)
  arr(1)=' '
  arrrr(1)=' '
  loop n(1,DOFL)
   IF TYSUB = 3 THEN
    arr(1) =string(arr(1))+' '+string(BELU_Mindlin(BEN,n))
   else
    arr(1) =string(arr(1))+' '+string(BELU(BEN,n))
```

```
    ENDIF
    arrrr(1) =string(arrrr(1))+' '+string(BELT(BEN,n))
   end_loop
   oo = write(arr,1)
   arr(1)= 'Traction t:'
   oo=write(arr,1)
   oo = write(arrrr,1)
 end_loop
 oo=close
end
def writing2
;---------------------------------------
; Write Some of FISH Functions
;---------------------------------------
 oo = open(filename1,2,1)
 ar(1)='BELU_Flac at iteration NO.='+string(iter)+' / epsolon='+string(epselontolEL(iter))
 oo = write(ar,1)
 arr(1)=' '
 oo = write(arr,1)
 loop BEN (1,NELM)
  ar(1)='RESULTS OF ELEMENT No.'+string(BEN)
  oo = write(ar,1)
  arr(1)= 'BELU_Flac:'
  oo=write(arr,1)
  arr(1)=' '
  arrrr(1)=' '
  loop n(1,DOFL)
   arr(1) =string(arr(1))+' '+string(BELU_Flac(BEN,n,iter))
   arrrr(1) =string(arrrr(1))+' '+string(BELU(BEN,n))
  end_loop
  oo = write(arr,1)
  arr(1)= 'BELU:'
  oo=write(arr,1)
  oo = write(arrrr,1)
 end_loop
 oo=close
end



def applyFlacvelocity
```

```
;-----------------------------------------------------------------
; Apply velocity over the Flac3D  side of the interface
;-----------------------------------------------------------------
 loop BEN(1,NELM)
  Loop m(2,NDEL1)
   if IncdFlacT(BEN,m)#0 then
    gpnt=find_gp(IncdFlacT(BEN,m))
    idv = IncdFlacT(BEN,m)
    if CD=3 then
     vx = BELU_Flac(BEN,3*(m-1)-2,iter)/STEPNUMBER
     vy = BELU_Flac(BEN,3*(m-1)-1,iter)/STEPNUMBER
     vz = BELU_Flac(BEN,3*(m-1),iter)/STEPNUMBER
     COMMAND
     Apply xv @vx range id @idv
     Apply yv @vy range id @idv
     Apply zv @vz range id @idv
     END_COMMAND
    else
     gpnt1=find_gp(IncdFlacT1(BEN,m))
     idv1 = IncdFlacT1(BEN,m)
     vx = BELU_Flac(BEN,2*(m-1)-1,iter)/STEPNUMBER
     vz = BELU_Flac(BEN,2*(m-1),iter)/STEPNUMBER
     COMMAND
      Apply xv @vx range id @idv
      Apply zv @vz range id @idv
      Apply xv @vx range id @idv1
      Apply zv @vz range id @idv1
     END_COMMAND
     if SPS=1 then
      COMMAND
       Apply yv 0 range id @idv
       Apply yv 0 range id @idv1
      END_COMMAND
     end_if
    end_if
   end_if
  end_loop
 end_loop
end
def NextiterFlacvelocity2
;-------------------------------------------------------------------------
```

```
; Obtain the interface nodal velocity vector for the next iteraion
;                  [Second coupling method]
;-------------------------------------------------------------------------
 IF TYSUB = 3 THEN
 loop BEN(1,NELM)
  loop m(1,NDEL)
   loop n(1,DOF)
    IF CD=3 THEN
     BELU(BEN,3*m-2)= BELU_Mindlin(BEN,3*m-1)
     BELU(BEN,3*m-1)=-BELU_Mindlin(BEN,3*m)
     BELU(BEN,3*m)=-BELU_Mindlin(BEN,3*m-2)
    ELSE
     BELU(BEN,2*m-1)= BELU_Mindlin(BEN,2*m)
     BELU(BEN,2*m)=-BELU_Mindlin(BEN,2*m-1)
    ENDIF
   endloop
  endloop
 endloop
 END_IF
 loop BEN(1,NELM)
  loop m(1,DOFL)
   BELU_Flac(BEN,m,iter+1)=float(BELU(BEN,m))
  endloop
 endloop
end
def NextiterFlacvelocity1
;-------------------------------------------------------------------------
; Obtain the interface nodal velocity vector for the next iteraion
;                  [First coupling method]
;-------------------------------------------------------------------------
 if TYSUB = 3 then
  loop BEN(1,NELM)
   loop m(1,NDEL)
    IF CD=3 THEN
     BELU(BEN,3*m-2)=BELU_Mindlin(BEN,3*m-1)
     BELU(BEN,3*m-1)=-BELU_Mindlin(BEN,3*m)
     BELU(BEN,3*m)=-BELU_Mindlin(BEN,3*m-2)
    ELSE
     BELU(BEN,2*m-1)=BELU_Mindlin(BEN,2*m)
     BELU(BEN,2*m)=-BELU_Mindlin(BEN,2*m-1)
    ENDIF
```

```
    endloop
   endloop
  endif
  loop BEN(1,NELM)
   Loop m(1,DOFL)
   BELU_Flac(BEN,m,iter+1)=((1-relaxation)*BELU_Flac(BEN,m,iter)+relaxation*BELU(BEN,m))
   end_loop
  end_loop
  nominator= 0
  denominator= 0
  loop BEN(1,NELM)
   loop m(1,DOFL)
    nominator= nominator+(BELU_Flac(BEN,m,iter+1)-BELU_Flac(BEN,m,iter))*(BELU_Flac(BEN,m,iter+1)-
    BELU_Flac(BEN,m,iter))
    denominator=denominator+BELU_Flac(BEN,m,iter+1)*BELU_Flac(BEN,m,iter+1)
   end_loop
  end_loop
  epselontolEL(iter)=sqrt(nominator/denominator)
  IF iter > 1
   IF epselontolEL(iter) < iteration_Tolerance THEN
   ;IF epselontolEL(iter-1) < epselontolEL(iter) THEN
   iterating= 1
   END_IF
  END_IF
end
def NextiterFlacvelocity3
;-------------------------------------------------------------------------
; Obtain the interface nodal velocity vector for the next iteraion
;    [First coupling method with different convergence condition]
;-------------------------------------------------------------------------
  if TYSUB = 3 then
   loop BEN(1,NELM)
    loop m(1,NDEL)
     IF CD=3 THEN
      BELU(BEN,3*m-2)=BELU_Mindlin(BEN,3*m-1)
      BELU(BEN,3*m-1)=-BELU_Mindlin(BEN,3*m)
      BELU(BEN,3*m)=-BELU_Mindlin(BEN,3*m-2)
     ELSE
      BELU(BEN,2*m-1)=BELU_Mindlin(BEN,2*m)
      BELU(BEN,2*m)=-BELU_Mindlin(BEN,2*m-1)
     ENDIF
```

```
    endloop
   endloop
  endif
  loop BEN(1,NELM)
   loop m(1,DOFL)
    BELU_Flac(BEN,m,iter+1)=((1-relaxation)*BELU_Flac(BEN,m,iter)+relaxation*BELU(BEN,m))
   end_loop
  end_loop
  Tempo= 0
  loop BEN(1,NELM)
   loop m(1,DOFL)
    Tempo=Tempo+(BELU_Flac(BEN,m,iter)-BELU(BEN,m))*(BELU_Flac(BEN,m,iter)-BELU(BEN,m))
   end_loop
  end_loop
  epselontolEL(iter)=sqrt(Tempo)
  ERFPW = (1.0/iter)
  ERF = (epselontolEL(iter) / epselontolEL(1))^(ERFPW)
  IF iter > 1
   if epselontolEL(iter) < Factorepsolon * epselontolEL(1) THEN
    NIT =iter
   end_if
   IF epselontolEL(iter) < iteration_Tolerance THEN
   ;IF epselontolEL(iter-1) < epselontolEL(iter) THEN
    iterating = 1
   END_IF
  END_IF
end
def ApplyFlacforce
;---------------------------------------------------------------------
; Obtain the interface nodal forces vector
;---------------------------------------------------------------------
  LOOP m(1,TNODDOF)
   BELf_Flac(m)=0
  END_LOOP
  loop BEN(1,NELM)
   Loop m(2,NDEL1)
   IF IncdFlacT(BEN,m)#0 THEN
    gpnt=find_gp(IncdFlacT(BEN,m))
    IF CD=3 Then
     BELf_Flac(3*Incd(BEN,m-1)-2)=gp_xfunbal(gpnt)+gp_xfapp(gpnt)
     BELf_Flac(3*Incd(BEN,m-1)-1)=gp_yfunbal(gpnt)+gp_yfapp(gpnt)
```

```
     BELf_Flac(3*Incd(BEN,m-1))=gp_zfunbal(gpnt)+gp_zfapp(gpnt)
    ELSE
     BELf_Flac(2*Incd(BEN,m-1)-1)=(2/len2)*(gp_xfunbal(gpnt)+gp_xfapp(gpnt))
     BELf_Flac(2*Incd(BEN,m-1))=(2/len2)*(gp_zfunbal(gpnt)+gp_zfapp(gpnt))
    ENDIF
   ENDIF
   end_loop
  end_loop
  IF TYSUB = 3 THEN
   LOOP m(1,TNOD)
    IF CD=3 Then
     BELf_FlacMindlin(3*m-1)=BELf_Flac(3*m-2)
     BELf_FlacMindlin(3*m) =-BELf_Flac(3*m-1)
     BELf_FlacMindlin(3*m-2)=-BELf_Flac(3*m)
    ELSE
     BELf_FlacMindlin(2*m)=BELf_Flac(2*m-1)
     BELf_FlacMindlin(2*m-1) =-BELf_Flac(2*m)
    ENDIF
   END_LOOP
  ENDIF
end
def Convergence1
;-----------------------------------------------------------------
; First Method Coupling Iteration Loop
;-----------------------------------------------------------------
 loop BEN(1,NELM)
  Loop m(1,DOFL)
   BELU_Flac(BEN,m,1)=0 ; apply initial velocity on Flac sub-domain interface nodes
  end_loop
 end_loop
 SECTION
  loop iter(1,iterations)
   Command
    ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress sxy 0.0 sxz 0.0 syz 0.0
    ini xvel 0.0 yvel 0.0 zvel 0.0
    ini xdis 0.0 ydis 0.0 zdis 0.0
    @applyFlacvelocity
    solve step @STEPNUMBER
   EndCommand
   ApplyFlacforce
   IF iter =1 THEN
```

```
     Input_INT
    ENDIF
    msg = 'iter='+string(iter)
    dum = out(msg)
    msg = 'Please wait for the BEM-C++ intrinsic /example_PreProcessing/ to excute its computations '
    dum = out(msg)
    if TYSUB = 3
     Pre = example_PreProcessing(CD,DOF,TYSUB,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,KelMind,SPS,
     NCCrd_Mindlin,Incd,BELf_FlacMindlin,BELU_Mindlin,BELT)
    else
     Pre = example_PreProcessing(CD,DOF,TYSUB,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,KelMind,SPS,
     NCCrd,Incd,BELf_Flac,BELU,BELT)
    end_if
    NextiterFlacvelocity3
    msg = ' NIT = '+ string (NIT)
    dum = out(msg)
    msg =' ERF = ' + string (ERF)
    dum = out(msg)
    table(101,iter) = NIT
    table(102,iter) = ERF
    IF iterating = 1 THEN
    ;if iter =iterations then
     writing1
     writing2
     EXIT SECTION
    END_IF
   end_loop
 END_SECTION
end
@Convergence1
def Convergence2
;-----------------------------------------------------------------
; Second Method Coupling Iteration Loop
;-----------------------------------------------------------------
 loop BEN(1,NELM)
  Loop m(1,DOFL)
   BELU_Flac(BEN,m,1)=0 ; apply initial velocity to Flac sub-domain interface nodes
  end_loop
 end_loop
 loop iter(1,2)
  Command
```

```
    ini sxx @xinitialstress syy @yinitialstress szz @zinitialstress sxy 0.0 sxz 0.0 syz 0.0
    ini xvel 0.0 yvel 0.0 zvel 0.0
    ini xdis 0.0 ydis 0.0 zdis 0.0
    @applyFlacvelocity
    solve step @STEPNUMBER
   EndCommand
   IF iter=1 Then
    Input_INT
    ApplyFlacforce
    msg = 'Please wait for the BEM-C++ intrinsinc /example_PreProcessing/ to excute its computations '
    dum = out(msg)
    if TYSUB = 3
     Pre = example_PreProcessing(CD,DOF,TYSUB,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,KelMind,SPS,
     NCCrd_Mindlin,Incd,BELf_FlacMindlin,BELU_Mindlin,BELT)
    else
     Pre = example_PreProcessing(CD,DOF,TYSUB,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,KelMind,SPS,
     NCCrd,Incd,BELf_Flac,BELU,BELT)
    end_if
    NextiterFlacvelocity2
    writing1
   ENDIF
 end_loop
end
;@Convergence2
@test1
list @tim
@nastr
@nadis
Pause
def PostProcess
;-------------------------------------------------------------------------
; Obtain the stress and the displacement at chosen internal points
; and the stress on the boundary (interface)
;-------------------------------------------------------------------------
 msg = 'Please wait for the BEM-C++ intrinsinc /example_PostProcessing/ to excute its computations '
 dum = out(msg)
 if TYSUB = 3
  Post
  =example_PostProcessing(CD,DOF,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,NELI,NELF,INTP,SPS,TYSUB,KelMin
  d,NCCrd_Mindlin,Incd,BELU_Mindlin,BELT,NCCrd_INT,StressBE,U_INT,Stress_INT)
 else
```

```
 Post =
 example_PostProcessing(CD,DOF,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,NELI,NELF,INTP,SPS,TYSUB,KelMind
 ,NCCrd,Incd,BELU,BELT,NCCrd_INT,StressBE,U_INT,Stress_INT)
end_if
oo = open(filename1,2,1)
arr(1)=' ------------------------------------------------'
oo=write(arr,1)
arr(1)=' Post-processed Results '
oo=write(arr,1)
arr(1)=' ------------------------------------------------'
oo=write(arr,1)
if NELI > 0 Then
 arr(1)='Results at Boundary Elements:'
 oo=write(arr,1)
 arr(1)= ' '
 oo=write(arr,1)
 loop BEN (1,NELF)
  loop Nodd (1,NDEL)
   arr(1)=' '
   arr(1) =string(arr(1))+'Element No.'+string(BEN)+' Global Node No. = '+string(Incd(BEN,Nodd))+' Local
   No./'+string(Nodd)+'/'
   oo = write(arr,1)
   ar(1)=' '
   loop m (1,VCD)
    ar(1) =string(ar(1))+' '+ string(StressBE(BEN,Nodd,m))
   end_loop
   oo = write(ar,1)
  end_loop
 end_loop
end_if
arr(1)=' '
oo = write(arr,1)
arr(1)='Internal Results:'
oo = write(arr,1)
arr(1)=' '
oo = write(arr,1)
loop intm(1,INTP)
 arr(1)='Coordinates'
 oo = write(arr,1)
 ar(1)=' '
 loop s (1,CD)
```

```
    ar(1) =string(ar(1))+' '+string(NCCrd_INT(intm,s))
   end_loop
   oo = write(ar,1)
   arr(1)=' displacement u: '
   oo = write(arr,1)
   ar(1)=' '
   loop ss (1,CD)
    ar(1) =string(ar(1))+' '+string(U_INT(intm,ss))
   end_loop
   oo = write(ar,1)
   arr(1)=' Stress: '
   oo = write(arr,1)
   ar(1)=' '
   loop sss (1,VCD)
    ar(1) =string(ar(1))+' '+string(Stress_INT(intm,sss))
   end_loop
   oo = write(ar,1)
 end_loop
 oo=close
end
@PostProcess
@test1
list @tim
pause
;-------------------------------------------------------------------------
; exact and numerical solution for a spherical Excavation problem [7]
; in BEM infinite sub-domain
;
; stores in
; Table 5: analytical values sigzzeU
; Table 6: numerical values sigzzU
; at zone centroid closed to x axis
;
; Table 7: errzrU the error in sigzzU
; Table 8: numerical values of ux (dispU) at grid points along x axis
;-------------------------------------------------------------------------
def Poststt
 command
  tab 5 name analytic-sigzzU
  tab 6 name BEM-sigzzU
  tab 7 name ERR-in-SigzzU
```

```
  tab 8 name BEM-dispU
 end_command
 loop i(1,INTP)
  r= sqrt(float(NCCrd_INT(i,1))^2 +float(NCCrd_INT(i,2))^2+float(NCCrd_INT(i,3))^2)
  A=(4-ny_Poisson*5)*(rad^3)
  B=2*(7-ny_Poisson*5)*(r^3)
  D=4.5*(rad^5)
  E=(7-5*ny_Poisson)*(r^5)
  sigzzeU = p*(1.+(A/B)+(D/E))
  sigzzU = float(Stress_INT(i,3))+p
  errzrU = ((sigzzU - sigzzeU)/sigzzeU)*100
  dispU = float(U_INT(i,1))
  table(8,r) = dispU
  table(5,r) = sigzzeU
  table(6,r) = sigzzU
  table(7,r) = errzrU
 end_loop
 A=(4-ny_Poisson*5)*(rad^3)
 B=2*(7-ny_Poisson*5)*(len^3)
 D=4.5*(rad^5)
 E=(7-5*ny_Poisson)*(len^5)
 sigzzUB = StressBE(BENiter,miter,3)+ p
 sigzzeUB = p*(1.+(A/B)+(D/E))
 errzrUB = ((sigzzUB - sigzzeUB)/sigzzeUB)*100
 table(5,len) = sigzzeUB
 table(6,len) = sigzzUB
 table(7,len) = errzrUB
end
@Poststt
plot create view verticalstresses
plot add table 1 , 2
title 'vertical stresses'
plot create view displacements
plot add table 4
title 'Numerical X Displacements'
plot create view verticalstressesU
plot add table 5 , 6
title 'vertical stresses in Unbounded Domain'
plot create view verticalstressesT
plot add table 1 , 2 , 5 , 6
title 'vertical stresses in both domains'
```

```
plot create view HorizontaldispU
plot add table 8
title 'Horizontal displacement in Unbounded Domain'
plot create view HorizontaldispT
plot add table 8, 4
title 'Horizontal displacement in both domains'
plot hist 2
```

```
;-------------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------------
; Numerical solution for a surface uniform load in semi-infinite Elastic medium {Holl Solution}
;                                  //Coupled Solution//
;-------------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------------
def Element_nunmber
;-------------------------------------------------------------------
; Obtain The Boundary Elements Number NELM [1]
;-------------------------------------------------------------------
 zpnt = zone_head
 NELM=0
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+gp_ypos(gpnt)^2+gp_zpos(gpnt)^2)
   if distance >= d-0.001 then
    mark = mark+1
   end_if
  end_loop
  if mark = 4 THEN
   NELM = NELM+1
  end_if
  zpnt = z_next(zpnt)
 end_loop
end
@Element_nunmber
def main_input
;-------------------------------------------------------
; Assign the basic information[2]
;-------------------------------------------------------
 array ar(1) arr(1) arrr(3,4) arrrr(1)
 filename1 = 'result.dat'
 oo = open(filename1,1,1)
 arr(1)='Project: Uniform Surface Load in Semi-infinte Medium'
 oo=write(arr,1)
 CD = 3                          ; Cartesian dimension (2D & 3D )
 arr(1)='Cartesian_dimension ='+string(CD)
 oo=write(arr,1)
 ELD= CD-1                      ; Boundary Element dimension
```

```
DOF = 3 ; Degrees of freedom
TYSUB = 3                    ; Type of Sub-domain (Finite BEM Sub-domain = 1, Infinite BEM Sub-domain =
2,Semi-Infinite BEM Sub-domain = 3)
KelMind= 1                   ; Method of Analysis for Semi-infinte Sub-domain or Type of Used Fundamental
solutions( Kelvin's = 0, Mindlin's = 1, Melan's = 2)
IF TYSUB = 1 THEN
 arr(1)='Finite Region'
 oo=write(arr,1)
ELSE
 IF TYSUB = 2 THEN
  arr(1)='Infinite Region'
  oo=write(arr,1)
 ELSE
  arr(1)='Semi-Infinite Region'
  oo=write(arr,1)
  IF KelMind=0
   arr(1)='Semi-Infinite Region/ Kelvin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=1
   arr(1)='Semi-Infinite Region/Mindlin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=2
   arr(1)='Semi-Infinite Region/Melan solution '
   oo=write(arr,1)
  END_IF
 END_IF
END_IF
TYM = 2           ; Symmetry about yz = 1, Symmetry about yz and xz = 2, Symmetry about yz, xz and xy =3)
CASEOF TYM
 arr(1)='No symmetry'
 oo=write(arr,1)
CASE 1
 arr(1)='Symmetry about y-z plane'
 oo=write(arr,1)
CASE 2
 arr(1)='Symmetry about y-z and x-z planes'
 oo=write(arr,1)
CASE 3
 arr(1)='Symmetry about all planes'
```

```
  oo=write(arr,1)
 ENDCASE
 SPS= 1
 ELTY=1              ; Element type
 ; Determine number of nodes per element NDEL
 IF CD = 2 THEN
  NDEL= 2
 ELSE
  NDEL= 4
 END_IF
 arr(1)=' Single Element Nodes number = '+ string(NDEL)
 oo=write(arr,1)
 ; Material Properties
 arr(1)=' Young modulus = '+string(E_Young)
 oo=write(arr,1)
 arr(1)=' Poisson ratio= '+string(ny_Poisson)
 oo=write(arr,1)
 iterations = 50               ; No. of iterations
 iterations1=iterations+1
 arr(1)=' Number of Iterations= '+string(iterations)
 oo=write(arr,1)
 relaxation=0.4
 arr(1)=' relaxation Parameter= '+string(relaxation)
 oo=write(arr,1)
 iteration_Tolerance = 0.0005    ; iteration tolerance should be higher if function NextiterFlacvelocity1
 is used (0.002)
 arr(1)=' iteration Tolerance= '+string(iteration_Tolerance)
 oo=write(arr,1)
 Factorepsolon = 0.01
 VCD= 2*CD                   ; No. of Stress Components at a Node
 DOFL= NDEL*DOF              ; Total degrees of freedom of element
 INTP = 35                   ; Number of internal points displacement and stress are required to be computed
 at
 NELI= 1
 NELF= NELM
 NDEL1= NDEL+1
 oo = close
end
@main_input
```

```
def Temporaryarray
;-------------------------------------------------------------------
; Obtain Boundary Elements Temporary Incidences Array[3]
;-------------------------------------------------------------------
 Array BEL(NELM)
 Array IncdTemp(NELM,NDEL)
 zpnt = zone_head
 BEN=1
 loop while zpnt # null
  mark = 0
  N=0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+gp_ypos(gpnt)^2+gp_zpos(gpnt)^2)
   if BEN <= NELM Then
    if distance >= d-0.001 then
     N=N+1
     IncdTemp(BEN,N) = gp_id(gpnt)
     TNODTempf=Max(TNODTempf,IncdTemp(BEN,N))
     mark =mark+1
    end_if
   end_if
  end_loop
  if mark = 4 THEN
   BEL(BEN)=z_id(zpnt)
   NELMTemp=Max(NELMTemp,BEL(BEN))
   BEN = BEN + 1
  end_if
  zpnt = z_next(zpnt)
 end_loop
end
@Temporaryarray
```

; Note: Fish functions BoundaryArrayFlac and BoundaryIncidences used in this semi-infinite 3D solution are the same functions used before in the infinite 3D solution, as shown in this Appendix g in p.411 and pp.414-417, respectively.

```
def NodesCoordinates
;-------------------------------------------------------------------------
; Compute Boundary Nodes Coordinates [6]
;-------------------------------------------------------------------------
```

```
array NCCrd(CD,TNOD)       ; Boundary Node co-ordinates
array NCCrd_Mindlin(CD,TNOD)
LOOP BEN(1,NELM)
 LOOP m(2,NDEL1)
  LOOP n (1,CD)
  NCCrd(n,Incd(BEN,m-1))= NCCrd_Flac(n,IncdFlacT(BEN,m))
  if abs(NCCrd(n,Incd(BEN,m-1))) < 10e-10 then
   NCCrd(n,Incd(BEN,m-1)) = 0.0
  end_if
  if NCCrd(3,Incd(BEN,m-1))= -d
   if NCCrd(2,Incd(BEN,m-1))=0
    if NCCrd(1,Incd(BEN,m-1))=0
     BENiter = BEN
     miter = m-1
    end_if
   end_if
  end_if
  END_LOOP
 END_LOOP
END_LOOP
IF TYSUB = 3 THEN
 LOOP m(1,TNOD)
  IF CD=3 THEN
   NCCrd_Mindlin(2,m) = NCCrd(1,m)
   NCCrd_Mindlin(1,m) =-NCCrd(3,m)
   NCCrd_Mindlin(3,m) =-NCCrd(2,m)
  ELSE
   NCCrd_Mindlin(2,m) = NCCrd(1,m)
   NCCrd_Mindlin(1,m) =-NCCrd(2,m)
  END_IF
 END_LOOP
END_IF
end
@NodesCoordinates
def Poststt
;-----------------------------------------------------------------------------------------------------
; exact and numerical solution for a uniform surface load in BEM Semi-infinte sub-domain [7]
; Table 22: analytical values AsigzzeU
; Table 23: numerical values AdiszzU
; Table 25: analytical values AsigxxeU
; Table 26: numerical values AsigxxU
```

```
; Table 24: AerrzrU error in AdiszzU
; Table 27: AerrxrU error in AdisxxU
; at zone centroid under the center of the load area
;
; Table 28: analytical values BsigzzeU
; Table 29: numerical values BdiszzU
; Table 31: analytical values BsigxxeU
; Table 32: numerical values BsigxxU
; Table 34: analytical values BsigxzeU
; Table 35: numerical values BsigxzU
; Table 30: BerrzrU error in BdiszzU
; Table 33: BerrxrU error in BdisxxU
; Table 35: BerrxzrU error in BsigxzU
; at zone centroid under the corner of the load area
;
; Table 37: analytical values Adisze
; Table 38: numerical values Adisz
; Table 39: errdisz_center error in Adisz
; at grid points under the center of the load area
; Table 40: analytical values Bdisze
; Table 41: numerical values Bdisz
; Table 42: errdisz_corner error in Bdisz
; at grid points under the corner of the load area
;-------------------------------------------------------------------------------------------------
 Num =0
 command
  tab 22 name Analytic-Center-sigzzU
  tab 23 name BEM-Center-sigzzU
  tab 24 name ERR-Center-in-SigzzU
  tab 25 name Analytic-Center-sigxxU
  tab 26 name BEM-Center-sigxxU
  tab 27 name ERR-Center-in-SigxxU
  tab 28 name Analytic-Corner-sigzzU
  tab 29 name BEM-Corner-sigzzU
  tab 30 name ERR-Corner-in-SigzzU
  tab 31 name Analytic-Corner-sigxxU
  tab 32 name BEM-Corner-sigxxU
  tab 33 name ERR-Corner-in-SigxxU
  tab 34 name Analytic-Corner-sigxzU
  tab 35 name BEM-Corner-sigxzU
  tab 36 name ERR-Corner-in-SigxzU
```

```
  tab 37 name Analytic-Center-diszU
  tab 38 name BEM-Center-diszU
  tab 39 name ERR-Center-in-diszU
  tab 40 name Analytic-Corner-diszU
  tab 41 name BEM-Corner-diszU
  tab 42 name ERR-Corner-in-diszU
 end_command
 loop i(1,INTP)
  global Depth =float(NCCrd_INT(i,1))
  if float(NCCrd_INT(i,2)) = 0 then
   global me = abs(XLength/YLength)
   global ne = abs(Depth/YLength)
   local Ae = sqrt(me^2+ne^2+1)
   local Be = me^2+2*ne^2+1
   local Ce = me^2+ne^2
   local De = 1+ne^2
   global Ie=(2/pi)*((me*ne/Ae)*(Be/(Ce*De))+ASIN(me/(sqrt(Ce)*sqrt(De))))
   sigzzee= -Ie*Load
   local R1=sqrt(XLength^2+Depth^2)
   local R2=sqrt(YLength^2+Depth^2)
   local R3=sqrt(XLength^2+YLength^2+Depth^2)
   local Iz5=(4/(2*pi))*(ATAN((-XLength*YLength)/(Depth*R3))+((-
   XLength*YLength*Depth/R3)*((1/R1^2)+(1/R2^2))))
   local Ix5=(1/(pi))*(ATAN((-XLength*YLength)/(Depth*R3))-((-2*XLength*YLength*Depth)/(R1^2*R3)))
   AsigzzeU= -Iz5*Load
   AsigxxeU= -Ix5*Load
   AsigzzU = -(float(Stress_INT(i,1))+zinitialstress)
   AerrzrU = ((AsigzzU - AsigzzeU)/(AsigzzeU))*100
   AsigxxU = -(float(Stress_INT(i,2))+xinitialstress)
   AerrxrU = ((AsigxxU - AsigxxeU)/(AsigxxeU))*100
   table(22,AsigzzeU)= -Depth
   table(23,AsigzzU) = -Depth
   table(24,-Depth) = AerrzrU
   table(25,AsigxxeU)= -Depth
   table(26,AsigxxU) = -Depth
   table(27,-Depth) = AerrxrU
   local n1 = Depth/XLength
   local m1 = XLength/(-YLength)
   local s1 = sqrt(1+m1^2+n1^2)
   local A =(1/(2*pi))*(ln((s1+m1)/(s1-m1))+m1*ln((s1+1)/(s1-1)))
   local B =(n1/(2*pi))*atan(m1/(n1*s1))
```

```
  local nyp = (1-2*ny_Poisson)/(1-ny_Poisson)
  Adisze = (4*Load*XLength)/(E_Young)*(1- ny_Poisson^2)*(A-nyp*B);
  Adisz = -float(U_INT(i,1))
  errdisz_center = ((Adisz - Adisze)/abs(Adisze))*100
  table(37,Adisze) = -Depth
  table(38,Adisz) = -Depth
  table(39,errdisz_center) = -Depth
 else
  R1=sqrt(XLength1^2+Depth^2)
  R2=sqrt(YLength1^2+Depth^2)
  R3=sqrt(XLength1^2+YLength1^2+Depth^2)
  local
  Iz4=(1/(2*pi))*(ATAN((XLength1*YLength1)/(Depth*R3))+((XLength1*YLength1*Depth/R3)*((1/R1^2)+(1/R2^2))))
  local Ix4=(1/(4*pi))*(ATAN((XLength1*YLength1)/(Depth*R3))-((2*XLength1*YLength1*Depth)/(R1^2*R3)))
  local Ixz4=(-1/(2*pi))*(((XLength1)/(R2))-((XLength1*Depth^2)/(R1^2*R3)))
  BsigzzeU= -Iz4*Load
  BsigxxeU= -Ix4*Load
  BsigxzeU= Ixz4*Load
  BsigzzU = -(float(Stress_INT(i,1))+zinitialstress)
  BerrzrU = ((BsigzzU - BsigzzeU)/(BsigzzeU))*100
  BsigxxU = -(float(Stress_INT(i,2))+xinitialstress)
  BerrxrU = ((BsigxxU - BsigxxeU)/(BsigxxeU))*100
  BsigxzU = -(float(Stress_INT(i,4))+xzinitialstress)
  BerrxzrU = ((BsigxzU - BsigxzeU)/(BsigxzeU))*100
  table(28,BsigzzeU)= -Depth
  table(29,BsigzzU) = -Depth
  table(30,-Depth) = BerrzrU
  table(31,-Depth)= BsigxxeU
  table(32,-Depth) = BsigxxU
  table(33,-Depth) = BerrxrU
  table(34,BsigxzeU)= -Depth
  table(35,BsigxzU) = -Depth
  table(36,-Depth) = BerrxzrU
  Num = Num +1
  n1 = Depth/XLength1
  m1 = XLength1/YLength1
  s1 = sqrt(1+m1^2+n1^2)
  A =(1/(2*pi))*(ln((s1+m1)/(s1-m1))+m1*ln((s1+1)/(s1-1)))
  B =(n1/(2*pi))*atan(m1/(n1*s1))
  nyp = (1-2*ny_Poisson)/(1-ny_Poisson)
  Bdisze = ((Load*XLength1)/(E_Young))*(1- ny_Poisson^2)*(A-nyp*B);
```

```
    Bdisz = -float(U_INT(i,1))
    errdisz_corner = ((Bdisz - Bdisze)/abs(Bdisze))*100
    table(40,Bdisze) = -Depth
    table(41,Bdisz) = -Depth
    table(42,errdisz_corner) = -Depth
   end_if
  end_loop
  Depth = len1
  R1 =sqrt(XLength^2+Depth^2)
  R2 =sqrt(YLength^2+Depth^2)
  R3 =sqrt(XLength^2+YLength^2+Depth^2)
  I4 =(4/(2*pi))*(ATAN((-XLength*YLength)/(Depth*R3))+((-XLength*YLength*Depth/R3)*((1/R1^2)+(1/R2^2))))
  sigzzeUB = -I4*Load
  sigzzUB = -StressBE(BENiter,miter,1)+zinitialstress
  errzrUB = ((sigzzUB - sigzzeUB)/sigzzeUB)*100
  table(22,sigzzeUB)= -Depth
  table(23,sigzzUB) = -Depth
  table(24,-Depth) = errzrUB
  ZoneNo=nzone
  loop i(1,Num)
   x1 = xtable(31,i)
   y1 = ytable(31,i)
   x2 = xtable(32,i)
   y2 = ytable(32,i)
   xtable(31,i) = y1
   ytable(31,i) = x1
   xtable(32,i) = y2
   ytable(32,i) = x2
  end_loop
end
@Poststst
plot create view Stressz-Under-Load-Center
plot add table 10 , 11
plot create view zstressUcenter
plot add table 22 , 23
title 'vertical stresses in Unbounded Domain under the load center'
plot create view zstressesTCenter
plot add table 10 , 11 , 22 , 23
title 'vertical stresses under the center in both domains'
plot create view Stressx-Under-Load-Center
plot add table 13 , 14
```

```
plot create view xstressUcenter
plot add table 25 , 26
title 'Horizontal stresses in Unbounded Domain under the load center'
plot create view xstressesTCenter
plot add table 13 , 14 , 25 , 26
title 'vertical stresses under the center in both domains'
plot create view Uz-Under-Load-Center
plot add table 19 , 20
plot create view Uz-Under-Load-Center-Unbounded
plot add table 37 , 38
plot create view Uz-Center-both-domains
plot add table 19 , 20, 37 , 38
plot create view Stressz-Under-Load-Corner
plot add table 1 , 2
plot create view zstressUcorner
plot add table 28 , 29
title 'vertical stresses in Unbounded Domain under the load corner'
plot create view zstressesTcorner
plot add table 1 , 2 , 28 , 29
title 'vertical stresses under the corner in both domains'
plot create view Stressx-Under-Load-Corner
plot add table 4 , 5
plot create view xstressUcorner
plot add table 31 , 32
title 'Horizontal stresses in Unbounded Domain under the load corner'
plot create view xstressesTcorner
plot add table 4 , 5 , 31 , 32
plot create view Stressxz-Under-Load-Corner
plot add table 7 , 8
plot create view xzstressUcorner
plot add table 34 , 35
title 'Shear stresses in Unbounded Domain under the load corner'
plot create view xzstressesTcorner
plot add table 7 , 8 , 34 , 35
plot create view Uz-Under-Load-Corner
plot add table 16 , 17
plot create view Uz-Under-Load-corner-Unbounded
plot add table 40 , 41
plot create view Uz-corner-both-domains
plot add table 16 , 17, 40 , 41
plot hist 2
```

```
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
; Numerical solution for a circular tunnel in infinite elastic material
;                      (Theoretical solution by Lame)
;                          //Coupled Solution//
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------

def Element_nunmber
;---------------------------------------------------------------
; Obtain The Boundary Elements Number NELM [1]
;---------------------------------------------------------------
 zpnt = zone_head
 NELM=0
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+(gp_zpos(gpnt))^2)
   if distance >= len-0.001 then
    if gp_ypos(gpnt) <= 0.001 then
     mark = mark+1
    end_if
   end_if
  end_loop
  if mark = 2 THEN
   NELM = NELM+1
  end_if
  zpnt = z_next(zpnt)
 end_loop
end
@Element_nunmber
def main_input
;-----------------------------------------------------
; Assign the basic information[2]
;-----------------------------------------------------
 array ar(1) arr(1) arrr(3,4) arrrr(1)
 filename1 = 'result.dat'
 oo = open(filename1,1,1)
 arr(1)='Project: A circular tunnel in a infinite Plane'
 oo=write(arr,1)
```

```
CD = 2          ; Cartesian dimension (2D & 3D )
arr(1)='Cartesian_dimension ='+string(CD)
oo=write(arr,1)
ELD=CD-1        ; Boundary Element dimension
DOF =2          ; Degrees of freedom
SPS=1           ; For 2D Problems: solid plane strain problems = 1,solid plane stress problems = 2
IF SPS = 1 THEN
 arr(1)='Type of Analysis: Solid Plane Strain'
 oo=write(arr,1)
ELSE
 arr(1)='Type of Analysis: Solid Plane Stress'
 oo=write(arr,1)
END_IF
TYSUB =2      ; Type of Sub-domain (Finite BEM Sub-domain = 1, Infinite BEM Sub-domain = 2,Semi-Infinite
BEM Sub-domain = 3)
KelMind=0     ; Method of Analysis for Semi-infinte Sub-domain or Type of Used Fundamental solutions(
Kelvin's = 0, Mindlin's = 1, Melan's = 2)
IF TYSUB = 1 THEN
 arr(1)='Finite Region'
 oo=write(arr,1)
ELSE
 IF TYSUB = 2 THEN
  arr(1)='Infinite Region'
  oo=write(arr,1)
 ELSE
  arr(1)='Semi-Infinite Region'
  oo=write(arr,1)
  IF KelMind=0
   arr(1)='Semi-Infinite Region/ Kelvin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=1
   arr(1)='Semi-Infinite Region/Mindlin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=2
   arr(1)='Semi-Infinite Region/Melan solution '
   oo=write(arr,1)
  END_IF
 END_IF
END_IF
```

```
TYM =2        ; Symmetry about yz = 1, Symmetry about yz and xz = 2, Symmetry about yz, xz and xy =3)
CASEOF TYM
 arr(1)='No symmetry'
 oo=write(arr,1)
CASE 1
 arr(1)='Symmetry about y-z plane'
 oo=write(arr,1)
CASE 2
 arr(1)='Symmetry about y-z and x-z planes'
 oo=write(arr,1)
CASE 3
 arr(1)='Symmetry about all planes'
 oo=write(arr,1)
ENDCASE
ELTY=1 ; Element type
; Determine number of nodes per element NDEL
IF CD = 2 THEN
 NDEL= 2
ELSE
 NDEL= 4
END_IF
arr(1)=' Single Element Nodes number = '+ string(NDEL)
oo=write(arr,1)
; Material Properties
arr(1)=' Young modulus = '+string(E_Young)
oo=write(arr,1)
arr(1)=' Poisson ratio= '+string(ny_Poisson)
oo=write(arr,1)
iterations =  50  ; No. of iterations
iterations1=iterations+1
arr(1)=' Number of Iterations= '+string(iterations)
oo=write(arr,1)
relaxation=0.4
arr(1)=' relaxation Parameter= '+string(relaxation)
oo=write(arr,1)
iteration_Tolerance =0.00006 ; iteration tolerance could be higherer if function NextiterFlacvelocity1 is
used (0.0006)
arr(1)=' iteration Tolerance= '+string(iteration_Tolerance)
oo=write(arr,1)
Factorepsolon = 0.01
VCD= 2*CD      ; No. of Stress Components at a Node
```

```
 DOFL= NDEL*DOF ; Total degrees of freedom of element
 INTP = 15      ; Number of internal points displacement and stress are required to be computed at
 NELI=1
 NELF=NELM
 NDEL1=NDEL+1
 oo = close
end
@main_input
def Temporaryarray
;-------------------------------------------------------------------------
; Obtain Boundary Elements Temporary Incidences Array[3]
;-------------------------------------------------------------------------
 Array BEL(NELM)
 Array IncdTemp(NELM,NDEL)
 Array IncdTemp1(NELM,NDEL)
 zpnt = zone_head
 BEN=1
 loop while zpnt # null
  mark = 0
  N=0
  M=0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+(gp_zpos(gpnt))^2)
   if BEN <= NELM Then
    if distance >= len-0.001 then
     if gp_ypos(gpnt) <= 0.001 then
      N=N+1
      IncdTemp(BEN,N) = gp_id(gpnt)
      TNODTempf=Max(TNODTempf,IncdTemp(BEN,N))
      mark =mark+1
     else
      M=M+1
      IncdTemp1(BEN,M) = gp_id(gpnt)
      TNODTempf1=Max(TNODTempf1,IncdTemp1(BEN,M))
     end_if
    end_if
   end_if
  end_loop
  if mark = 2 THEN
   BEL(BEN)=z_id(zpnt)
```

```
    NELMTemp=Max(NELMTemp,BEL(BEN))
    BEN = BEN + 1
   end_if
   zpnt = z_next(zpnt)
 end_loop
end
@Temporaryarray
def BoundaryArrayFlac
;---------------------------------------------------------------------
; Obtain Boundary Nodes Temporary Coordinates Array [4]
;---------------------------------------------------------------------
 Array IncdFlac(NELM,NDEL1)
 Array IncdFlacT(NELM,NDEL1)
 Array IncdFlac1(NELM,NDEL1)
 Array IncdFlacT1(NELM,NDEL1)
 Array NCCrd_Flac(CD,TNODTempf)
 Array NCCrd_Flac1(CD,TNODTempf1)
 Array IncdLFlac(NDEL)
 loop BEN(1,NELM)
  IncdFlac(BEN,1)=BEL(BEN)
  IncdFlac1(BEN,1)=BEL(BEN)
  loop m(2,NDEL1)
   IncdFlac(BEN,m)=IncdTemp(BEN,m-1)
   IncdFlacT(BEN,m)=IncdFlac(BEN,m)
   IncdFlac1(BEN,m)=IncdTemp1(BEN,m-1)
   IncdFlacT1(BEN,m)=IncdFlac1(BEN,m)
  end_loop
 end_loop
 loop m(1,NELM)
  Loop n(2,NDEL1)
   IF IncdFlac(m,n)#0 THEN
    gpnt=find_gp(IncdFlac(m,n))
    NCCrd_Flac(1,IncdFlac(m,n))=gp_xpos(gpnt)
    NCCrd_Flac(2,IncdFlac(m,n))=gp_zpos(gpnt)
   END_IF
   IF IncdFlac1(m,n)#0 THEN
    gpnt1=find_gp(IncdFlac1(m,n))
    NCCrd_Flac1(1,IncdFlac1(m,n))=gp_xpos(gpnt1)
    NCCrd_Flac1(2,IncdFlac1(m,n))=gp_zpos(gpnt1)
   END_IF
  end_loop
```

```
 end_loop
end
@BoundaryArrayFlac
def BoundaryIncidences1
;--------------------------------------------------------------------
; Obtain the Boundary Elements Outward Normal and
; Their Incidences array [5]
;--------------------------------------------------------------------
 LOOP BEN(1,NELM)
  LOOP m(2,NDEL1)
   LOOP n (1,CD)
    CoordL(n,m-1)= NCCrd_Flac1(n,IncdFlac1(BEN,m))
   END_LOOP
  IncdLFlac(m-1)= IncdFlac1(BEN,m)
 END_LOOP
 LOOP SC (1,CD)
  v_Outnormal(SC) =0
 END_LOOP
 xsai_Outnormal=0
 etta_Outnormal=0
 Outnormal
 LOOP SC (1,CD)
  v_BIC(SC)= v_Outnormal(SC)
 END_LOOP
 Shape_Function
 Coordinate
 LOOP SC (1,CD)
  Coord_BIC(SC)= Coord(SC)
 END_LOOP
 Costh_BIC=0
 loop Sc (1,CD)
  Costh_BIC=Costh_BIC+ V_BIC(Sc)*Coord_BIC(Sc)
 end_loop
 IF CD = 3 THEN
  Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2+Coord_BIC(3)^2)
 ELSE
  Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2)
 END_IF
 Costh_BIC=Costh_BIC/Costh
 IF CD = 3 THEN
  IF Costh_BIC > 0 Then
```

```
    IncdFlac(BEN,2)= IncdFlacT(BEN,5)
    IncdFlac(BEN,3)= IncdFlacT(BEN,4)
    IncdFlac(BEN,4)= IncdFlacT(BEN,3)
    IncdFlac(BEN,5)= IncdFlacT(BEN,2)
   END_IF
  END_IF
  IF CD = 2 THEN
   IF Costh_BIC > 0 Then
    IncdFlac1(BEN,2)= IncdFlacT1(BEN,3)
    IncdFlac1(BEN,3)= IncdFlacT1(BEN,2)
   END_IF
  END_IF
  END_LOOP
  LOOP BEN(1,NELM)
   LOOP m(1,NDEL1)
    IncdFlacT1(BEN,m)= IncdFlac1(BEN,m)
   End_LOOP
  End_LOOP
end
@BoundaryIncidences1
def NodesCoordinates
;-------------------------------------------------------------------------
; Compute Boundary Nodes Coordinates [6]
;-------------------------------------------------------------------------
 array NCCrd(CD,TNOD)      ; Boundary Node co-ordinates
 array NCCrd_Mindlin(CD,TNOD)
 LOOP BEN(1,NELM)
  LOOP m(2,NDEL1)
   LOOP n (1,CD)
    NCCrd(n,Incd(BEN,m-1))= NCCrd_Flac(n,IncdFlacT(BEN,m))
    if abs(NCCrd(n,Incd(BEN,m-1))) < 10e-10 then
     NCCrd(n,Incd(BEN,m-1)) = 0.0
    end_if
    if NCCrd(1,Incd(BEN,m-1))= len
     if NCCrd(2,Incd(BEN,m-1))=0
      BENiter = BEN
      miter = m-1
     end_if
    end_if
   END_LOOP
  END_LOOP
```

```
 END_LOOP
 IF TYSUB = 3 THEN
  LOOP m(1,TNOD)
   IF CD=3 THEN
    NCCrd_Mindlin(2,m) = NCCrd(1,m)
    NCCrd_Mindlin(1,m) =-NCCrd(3,m)
    NCCrd_Mindlin(3,m) =-NCCrd(2,m)
   ELSE
    NCCrd_Mindlin(2,m) = NCCrd(1,m)
    NCCrd_Mindlin(1,m) =-NCCrd(2,m)
   END_IF
  END_LOOP
 END_IF
end
@NodesCoordinates
;-------------------------------------------------------------------------------------------------------
; exact and numerical solution for a Tunnel in the BEM unbounded sub-domain [7]
; stores in
; Table 10: analytical radial stress sigreU/p
; Table 12: numerical radial stress sigrU/p
; Table 14: errsrU the error in sigr
; Table 11: analytical tangential stress sigteU/p
; Table 13: numerical tangential stress sigtU/p
; at zone centroid closed to x axis
; Table 15: errstU the error in sigt
; Table 16: analytical radial displacement ur at grid points diseU
; Table 17: numerical radial displacement ur at grid points disU
; Table 18: errU the error in ur
;-------------------------------------------------------------------------------------------------------
def nastrU
 command
  tab 10 name analytic-sigrU
  tab 11 name analytic-sigtU
  tab 12 name BEM-sigrU
  tab 13 name BEM-sigtU
  tab 14 name Error-in-sigrU
  tab 15 name Error-in-sigtU
 end_command
 errsrU = 0.0
 errstU = 0.0
 loop i(1,INTP)
```

```
   radd= sqrt(float(NCCrd_INT(i,1))^2 +float(NCCrd_INT(i,2))^2)
   sigreU = (1.-(rad/radd)^2)
   sigteU = (1.+(rad/radd)^2)
   sigxxU = float(Stress_INT(i,1))+xinitialstress
   sigzzU = float(Stress_INT(i,2))+zinitialstress
   sigxzU = float(Stress_INT(i,4))
   aux1 = (sigxxU+sigzzU) * 0.5
   aux2 = sqrt(sigxzU^2 + 0.25 *( sigxxU- sigzzU)^2)
   sigrU = -(aux1 + aux2) / p
   sigtU = -(aux1 - aux2) / p
   errsrU = ((sigrU - sigreU)/sigreU)*100
   errstU = ((sigtU - sigteU)/sigteU)*100
   table(10,radd) = sigreU
   table(11,radd) = sigteU
   table(12,radd) = sigrU
   table(13,radd) = sigtU
   table(14,radd) = errsrU
   table(15,radd) = errstU
 end_loop
 sigreUB = (1.-(rad/len)^2)
 sigteUB = (1.+(rad/len)^2)
 sigxxUB = StressBE(BENiter,miter,1)+xinitialstress
 sigzzUB = StressBE(BENiter,miter,2)+zinitialstress
 sigxzUB = StressBE(BENiter,miter,4)
 aux1 = (sigxxUB+sigzzUB) * 0.5
 aux2 = sqrt(sigxzUB^2 + 0.25 *( sigxxUB- sigzzUB)^2)
 sigrUB = -(aux1 + aux2) / p
 sigtUB = -(aux1 - aux2) / p
 errsrUB = ((sigrUB - sigreUB)/sigreUB)*100
 errstUB = ((sigtUB - sigteUB)/sigteUB)*100
 table(10,len) = sigreUB
 table(11,len) = sigteUB
 table(12,len) = sigrUB
 table(13,len) = sigtUB
 table(14,len) = errsrUB
 table(15,len) = errstUB
end
def nadisU
 command
  tab 16 name analytic-ur-U
  tab 17 name BEM-ur-U
```

```
  tab 18 name Error-in-ur-U
 end_command
 loop i(1,INTP)
  radd= sqrt(float(NCCrd_INT(i,1))^2 +float(NCCrd_INT(i,2))^2)
  diseU =0.5*(p/G_kelvin)*(rad^2/radd)
  disU = sqrt(U_INT(i,1)^2 + U_INT(i,2)^2)
  errU = ((disU-diseU)/diseU)*100
  table(16,radd) = diseU
  table(17,radd) = disU
  table(18,radd) = errU
 end_loop
end
@nastr
@nadis
@nastrU
@nadisU
plot create view stresses
plot set name 'Theoretical and Numerical Radial and Hoop Stresses in Flac domain'
plot add table 1 ,3,2,4
plot create view displacements
plot set name 'Theoretical and Numerical Radial Displacements'
plot add table 7,8
plot create view stressesU
plot set name 'Theoretical and Numerical Radial and Hoop Stresses in Unbounded Domain'
plot add table 10 ,12,11,13
plot create view displacementsU
plot set name 'Theoretical and Numerical Radial Displacements in Unbounded Domain'
plot add table 16,17
plot create view stressesT
plot set name 'Theoretical and Numerical Radial and Hoop Stresses in both Domains'
plot add table 1,3,2,4,10,12,11,13
plot create view displacementsT
plot set name 'Theoretical and Numerical Radial Displacements in both Domains'
plot add table 7,8, 16,17
```

```
;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------
; Numerical solution for a Hole in a plate in Elastic material{First Configuration}
;                         (Theoretical solution by Mindlin)
;                              //Coupled Solution//
;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------
def Element_nunmber
;-------------------------------------------------------------------
; Obtain The Boundary Elements Number NELM
;-------------------------------------------------------------------
 zpnt = zone_head
 NELM=0
 loop while zpnt # null
  mark = 0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+(gp_zpos(gpnt)-len3)^2)
   if distance >= len1-0.001 then
    if gp_ypos(gpnt) >= -0.001 then
     mark = mark+1
    end_if
   end_if
  end_loop
  if mark = 2 THEN
   NELM = NELM+1
  end_if
  zpnt = z_next(zpnt)
 end_loop
end
@Element_nunmber
def main_input
;-------------------------------------------------------
; Assign the basic information
;-------------------------------------------------------
 array ar(1) arr(1) arrr(3,4) arrrr(1)
 filename1 = 'result.dat'
 oo = open(filename1,1,1)
 arr(1)='Project: A Hole in a Semi infinite Plate'
 oo=write(arr,1)
 CD = 2         ; Cartesian dimension (2D & 3D )
```

```
arr(1)='Cartesian_dimension ='+string(CD)
oo=write(arr,1)
ELD=CD-1        ; Boundary Element dimension
DOF =2          ; Degrees of freedom
SPS=2           ; For 2D Problems: solid plane strain problems = 1,solid plane stress problems = 2
IF SPS = 1 THEN
 arr(1)='Type of Analysis: Solid Plane Strain'
 oo=write(arr,1)
ELSE
 arr(1)='Type of Analysis: Solid Plane Stress'
 oo=write(arr,1)
END_IF
TYSUB =3        ; Type of Sub-domain (Finite BEM Sub-domain = 1, Infinite BEM Sub-domain = 2,Semi-Infinite
BEM Sub-domain = 3)
KelMind=2       ; Method of Analysis for Semi-infinte Sub-domain or Type of Used Fundamental solutions(
Kelvin's = 0, Mindlin's = 1, Melan's = 2)
IF TYSUB = 1 THEN
 arr(1)='Finite Region'
 oo=write(arr,1)
ELSE
 IF TYSUB = 2 THEN
  arr(1)='Infinite Region'
  oo=write(arr,1)
 ELSE
  arr(1)='Semi-Infinite Region'
  oo=write(arr,1)
  IF KelMind=0
   arr(1)='Semi-Infinite Region/ Kelvin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=1
   arr(1)='Semi-Infinite Region/Mindlin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=2
   arr(1)='Semi-Infinite Region/Melan solution '
   oo=write(arr,1)
  END_IF
 END_IF
END_IF
TYM =1          ; Symmetry about yz = 1, Symmetry about yz and xz = 2, Symmetry about yz, xz and xy =3)
```

```
CASEOF TYM
 arr(1)='No symmetry'
 oo=write(arr,1)
CASE 1
 arr(1)='Symmetry about y-z plane'
 oo=write(arr,1)
CASE 2
 arr(1)='Symmetry about y-z and x-z planes'
 oo=write(arr,1)
CASE 3
 arr(1)='Symmetry about all planes'
 oo=write(arr,1)
ENDCASE
ELTY=1          ; Element type
; Determine number of nodes per element NDEL
IF CD = 2 THEN
 NDEL= 2
ELSE
 NDEL= 4
END_IF
arr(1)=' Single Element Nodes number = '+ string(NDEL)
oo=write(arr,1)
; Material Properties
arr(1)=' Young modulus = '+string(E_Young)
oo=write(arr,1)
arr(1)=' Poisson ratio= '+string(ny_Poisson)
oo=write(arr,1)
iterations = 50    ; No. of iterations
iterations1=iterations+1
arr(1)=' Number of Iterations= '+string(iterations)
oo=write(arr,1)
relaxation=0.5
arr(1)=' relaxation Parameter= '+string(relaxation)
oo=write(arr,1)
iteration_Tolerance =0.0006  ; iteration tolerance should be lower if function NextiterFlacvelocity3 is
used (0.00006)
arr(1)=' iteration Tolerance= '+string(iteration_Tolerance)
oo=write(arr,1)
Factorepsolon = 0.01
VCD= 2*CD         ; No. of Stress Components at a Node
DOFL= NDEL*DOF    ; Total degrees of freedom of element
```

```
 INTP = 19          ; Number of internal points displacement and stress are required to be computed at
 NELI=1
 NELF=NELM
 NDEL1=NDEL+1
 oo = close
end
@main_input
def Temporaryarray
;---------------------------------------------------------------------
; Obtain Boundary Elements Temporary Incidences Array
;---------------------------------------------------------------------
 Array BEL(NELM)
 Array IncdTemp(NELM,NDEL)
 Array IncdTemp1(NELM,NDEL)
 zpnt = zone_head
 BEN=1
 loop while zpnt # null
  mark = 0
  N=0
  M=0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+(gp_zpos(gpnt)-len3)^2)
   if BEN <= NELM Then
    if distance >= len1-0.001 then
     if gp_ypos(gpnt) = 0 then
      N=N+1
      IncdTemp(BEN,N) = gp_id(gpnt)
      TNODTempf=Max(TNODTempf,IncdTemp(BEN,N))
      mark =mark+1
     else
      M=M+1
      IncdTemp1(BEN,M) = gp_id(gpnt)
      TNODTempf1=Max(TNODTempf1,IncdTemp1(BEN,M))
     end_if
    end_if
   end_if
  end_loop
  if mark = 2 THEN
  BEL(BEN)=z_id(zpnt)
  NELMTemp=Max(NELMTemp,BEL(BEN))
```

```
   BEN = BEN + 1
   end_if
   zpnt = z_next(zpnt)
  end_loop
end
@Temporaryarray

def BoundaryIncidences
;-------------------------------------------------------------------
; Obtain the Boundary Elements Outward Normal and
; Their Incidences Array
;-------------------------------------------------------------------
LOOP BEN(1,NELM)
 LOOP m(2,NDEL1)
  LOOP n (1,CD)
   CoordL(n,m-1)= NCCrd_Flac(n,IncdFlac(BEN,m))
  END_LOOP
  IncdLFlac(m-1)= IncdFlac(BEN,m)
 END_LOOP
 LOOP SC (1,CD)
  v_Outnormal(SC) =0
 END_LOOP
 xsai_Outnormal=0
 etta_Outnormal=0
 Outnormal
 LOOP SC (1,CD)
  v_BIC(SC)= v_Outnormal(SC)
 END_LOOP
 Shape_Function
 Coordinate
 LOOP SC (1,CD)
  Coord_BIC(SC)= Coord(SC)
 END_LOOP
 Costh_BIC=0
 loop Sc (1,CD)
  Costh_BIC=Costh_BIC+ V_BIC(Sc)*Coord_BIC(Sc)
 end_loop
 IF CD = 3 THEN
  Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2+Coord_BIC(3)^2)
 ELSE
  Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2)
```

```
 END_IF
 Costh_BIC=Costh_BIC/Costh
 IF CD = 3 THEN
  IF Costh_BIC > 0 Then
   IncdFlac(BEN,2)= IncdFlacT(BEN,5)
   IncdFlac(BEN,3)= IncdFlacT(BEN,4)
   IncdFlac(BEN,4)= IncdFlacT(BEN,3)
   IncdFlac(BEN,5)= IncdFlacT(BEN,2)
  END_IF
 END_IF
 IF CD = 2 THEN
 ; IF Costh_BIC > 0 Then
  IncdFlac(BEN,2)= IncdFlacT(BEN,3)
  IncdFlac(BEN,3)= IncdFlacT(BEN,2)
 ; END_IF
 END_IF
END_LOOP
LOOP BEN(1,NELM)
 LOOP m(1,NDEL1)
  IncdFlacT(BEN,m)= IncdFlac(BEN,m)
 End_LOOP
End_LOOP
I=0
LOOP BEN(1,NELM)
 LOOP m(2,NDEL1)
  I=I+1
  IncdFlac(BEN,m)= I
  IF BEN >1 THEN
   SECTION
    LOOP s(1,BEN-1)
     LOOP mm(2,NDEL1)
      IF IncdFlacT(s,mm)=IncdFlacT(BEN,m) THEN
       IncdFlac(BEN,m)=IncdFlac(s,mm)
       I=I-1
       EXIT SECTION
      END_IF
     End_LOOP
    End_LOOP
   END_SECTION
  END_IF
 End_LOOP
```

```
End_LOOP
loop BEN(1,NELM)
 loop m(2,NDEL1)
  Incd(BEN,m-1)=IncdFlac(BEN,m)
  TNOD=Max(TNOD,Incd(BEN,m-1))
 end_loop
end_loop
TNODDOF=TNOD*DOF
oo = open(filename1,2,1)
arr(1)='Number of Nodes of The System= '+string(TNOD)
oo=write(arr,1)
arr(1)='Number of Elements of The System= '+string(NELM)
oo=write(arr,1)
oo=close
end
@BoundaryIncidences
def BoundaryIncidences1
;---------------------------------------------------------------------
; Obtain the Boundary Elements Outward Normal and
; Their Incidences array
;---------------------------------------------------------------------
 LOOP BEN(1,NELM)
  LOOP m(2,NDEL1)
   LOOP n (1,CD)
    CoordL(n,m-1)= NCCrd_Flac1(n,IncdFlac1(BEN,m))
   END_LOOP
   IncdLFlac(m-1)= IncdFlac1(BEN,m)
  END_LOOP
  LOOP SC (1,CD)
   v_Outnormal(SC) =0
  END_LOOP
  xsai_Outnormal=0
  etta_Outnormal=0
  Outnormal
  LOOP SC (1,CD)
   v_BIC(SC)= v_Outnormal(SC)
  END_LOOP
  Shape_Function
  Coordinate
  LOOP SC (1,CD)
   Coord_BIC(SC)= Coord(SC)
```

```
   END_LOOP
   Costh_BIC=0
   loop Sc (1,CD)
    Costh_BIC=Costh_BIC+ V_BIC(Sc)*Coord_BIC(Sc)
   end_loop
   IF CD = 3 THEN
    Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2+Coord_BIC(3)^2)
   ELSE
    Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2)
   END_IF
   Costh_BIC=Costh_BIC/Costh
   IF CD = 3 THEN
    IF Costh_BIC > 0 Then
     IncdFlac(BEN,2)= IncdFlacT(BEN,5)
     IncdFlac(BEN,3)= IncdFlacT(BEN,4)
     IncdFlac(BEN,4)= IncdFlacT(BEN,3)
     IncdFlac(BEN,5)= IncdFlacT(BEN,2)
    END_IF
   END_IF
   IF CD = 2 THEN
   ; IF Costh_BIC > 0 Then
    IncdFlac1(BEN,2)= IncdFlacT1(BEN,3)
    IncdFlac1(BEN,3)= IncdFlacT1(BEN,2)
   ; END_IF
   END_IF
 END_LOOP
 LOOP BEN(1,NELM)
  LOOP m(1,NDEL1)
   IncdFlacT1(BEN,m)= IncdFlac1(BEN,m)
  End_LOOP
 End_LOOP
end
@BoundaryIncidences1

def NodesCoordinates
;-------------------------------------------------------------------------
; Compute Boundary Nodes Coordinates
;-------------------------------------------------------------------------
 array NCCrd(CD,TNOD) ; Boundary Node co-ordinates
 array NCCrd_Mindlin(CD,TNOD)
 LOOP BEN(1,NELM)
```

```
    LOOP m(2,NDEL1)
     LOOP n (1,CD)
      NCCrd(n,Incd(BEN,m-1))= NCCrd_Flac(n,IncdFlacT(BEN,m))
      if abs(NCCrd(n,Incd(BEN,m-1))) < 10e-10 then
       NCCrd(n,Incd(BEN,m-1)) = 0.0
      end_if
     END_LOOP
    END_LOOP
  END_LOOP
  IF TYSUB = 3 THEN
   LOOP m(1,TNOD)
    IF CD=3 THEN
     NCCrd_Mindlin(2,m) = NCCrd(1,m)
     NCCrd_Mindlin(1,m) =-NCCrd(3,m)
     NCCrd_Mindlin(3,m) =-NCCrd(2,m)
    ELSE
     NCCrd_Mindlin(2,m) = NCCrd(1,m)
     NCCrd_Mindlin(1,m) =-NCCrd(2,m)
    END_IF
   END_LOOP
  END_IF
end
@NodesCoordinates
def ApplyFlacforce
;--------------------------------------------------------------------
; Obtain the interface nodal forces vector
;--------------------------------------------------------------------
 LOOP m(1,TNODDOF)
  BELf_Flac(m)=0
 END_LOOP
 loop BEN(1,NELM)
  Loop m(2,NDEL1)
   IF IncdFlacT(BEN,m)#0 THEN
    gpnt=find_gp(IncdFlacT(BEN,m))
    IF CD=3 Then
     BELf_Flac(3*Incd(BEN,m-1)-2)=gp_xfunbal(gpnt)+gp_xfapp(gpnt)
     BELf_Flac(3*Incd(BEN,m-1)-1)=gp_yfunbal(gpnt)+gp_yfapp(gpnt)
     BELf_Flac(3*Incd(BEN,m-1))=gp_zfunbal(gpnt)+gp_zfapp(gpnt)
    ELSE
     BELf_Flac(2*Incd(BEN,m-1)-1)=(-2/len2)*(gp_xfunbal(gpnt)+gp_xfapp(gpnt))
     BELf_Flac(2*Incd(BEN,m-1))=(-2/len2)*(gp_zfunbal(gpnt)+gp_zfapp(gpnt))
```

```
     ENDIF
    ENDIF
   end_loop
  end_loop
  IF TYSUB = 3 THEN
   LOOP m(1,TNOD)
    IF CD=3 Then
     BELf_FlacMindlin(3*m-1)=BELf_Flac(3*m-2)
     BELf_FlacMindlin(3*m) =-BELf_Flac(3*m-1)
     BELf_FlacMindlin(3*m-2)=-BELf_Flac(3*m)
    ELSE
     BELf_FlacMindlin(2*m)=BELf_Flac(2*m-1)
     BELf_FlacMindlin(2*m-1) =-BELf_Flac(2*m)
    ENDIF
   END_LOOP
  ENDIF
end

def nastrU
;-------------------------------------------------------------------------
; BEM and Coupled solutions of a hole in a semi-infinite medium problem
; stores in
; Table 1: BEM normalised stress sigxxeU/Tension
; Table 2: Coupled normalised stress sigxxU/Tension
; Table 3: errxrU the error in sigxxU
;-------------------------------------------------------------------------
 array sigxxeU(INTP)
 sigxxeU(1)=0.97906
 sigxxeU(2)=0.98096
 sigxxeU(3)=0.98634
 sigxxeU(4)=0.99431
 sigxxeU(5)=1.00357
 sigxxeU(6)=1.01265
 sigxxeU(7)=1.02015
 sigxxeU(8)=1.02504
 sigxxeU(9)=1.02677
 sigxxeU(10)=1.02538
 sigxxeU(11)=1.02143
 sigxxeU(12)=1.01587
 sigxxeU(13)=1.00983
 sigxxeU(14)=1.00435
```

```
 sigxxeU(15)=1.00023
 sigxxeU(16)=0.99783
 sigxxeU(17)=0.99709
 sigxxeU(18)=0.99758
 sigxxeU(19)=0.99865
 command
  tab 1 name BEM-Edge-sigxxU
  tab 2 name Coupled-Edge-sigxxU
  tab 3 name ERR-Edge-SigxxU
 end_command
 loop i(1,INTP)
  sigxxU = (float(Stress_INT(i,2))+xinitialstress)/xinitialstress
  errxrU = ((sigxxU - sigxxeU(i))/(sigxxeU(i)))*100
  radd = float(NCCrd_INT(i,2))/rad
  table(1,radd) = sigxxeU(i)
  table(2,radd) = sigxxU
  table(3,radd) = errxrU
 end_loop
end
@nastrU
plot create view XstressU-straight-edge
plot add table 1,2
title 'X stress in Unbounded Domain at Free Traction Edge'
```

```
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Numerical solution for a smooth square footing on Tresca material problem
;                          -associated plastic flow-
;                            //Coupled Solution//
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
 New
 CONFIG cppfish
 LOAD function BEMKMM002_64.dll
 def testt
  global tim0=clock
 end
 def test1
  global tim=(clock-tim0)/100.0
 end
 @testt
 def parm
  global len = 9.0 ; length of outer box edge
  global in_size = 8 ; number of zones along outer cube edge
  global STEPNUMBER = 8000
  global d_a = 3.0
  global d_b = 3.0
  global rad = d_b
  global rab = d_b/d_a
  global eps = 0.1
  global ae = d_a + eps
  global be = d_b + eps
 end
 @parm
 gen zone radbrick size @in_size @in_size @in_size @in_size rat 1.0 1.0 1.0 1 p0 0 0 0 p1 @len 0 0 p2 0
 @len 0 &
 p3 0 0 @len p4 @len @len 0 p5 0 @len @len p6 @len 0 @len p7 @len @len @len dim @rad @rad @rad
 def make_sphere
 ; Loop over all GPs and remap their coordinates:
 ; assume len>rad
  p_gp=gp_head
  loop while p_gp#null
   ; Get gp coordinate: P=(px,py,pz)
   px=gp_xpos(p_gp)
   py=gp_ypos(p_gp)
```

```
    pz=gp_zpos(p_gp)
    maxp=max(abs(px),max(abs(py),abs(pz)))
     ;k=len/maxp
     ; Compute A=(ax,ay,az)=point on sphere radially "below" P.
    dist=sqrt(px*px+py*py+pz*pz)
    if dist> 0 then
     ;k=rad/dist
     k=rad/maxp
     ax=px*k
     ay=py*k
     az=pz*k
      ; Compute B=(bx,by,bz)=point on outer box boundary radially "above" P.
     maxp=max(abs(px),max(abs(py),abs(pz)))
     k=len/dist
     bx=px*k
     by=py*k
     bz=pz*k
     ; Linear interpolation: P=A+u*(B-A)
     u=(maxp-rad)/(len-rad)
     gp_xpos(p_gp)=ax+u*(bx-ax)
     gp_ypos(p_gp)=ay+u*(by-ay)
     gp_zpos(p_gp)=az+u*(bz-az)
    end_if
    p_gp=gp_next(p_gp)
  end_loop
end
@make_sphere
gen zone brick size @in_size @in_size @in_size rat 1.0 1.0 1.0 p0 0 0 0 p1 @rad 0 0 p2 0 @rad 0 p3 0 0
@rad
;gen zone reflect dip 90 dd 90 origin 0 0 0
;gen zone reflect dip 90 dd 180 origin 0 0 0
plot create view Brick
plot set name 'Brick'
plot add zone addlabel "Zone" yellow
plot boundary
plot add axes
model mech mohr
prop bul 2.e8 shea 1.e8 cohesion 1.e5
prop friction 0. dilation 0. tension 1.e10
;-----------------------------------------------------------------------
; p_load : average footing pressure / c
```

```
; p_solup : upper bound value for the bearing capacity / c
; p_sollo : lower bound value for the bearing capacity / c
; c_disp : vertical displacement at footing center / a
;-----------------------------------------------------------------------
def p_cons
 if rab > 0.53 then
  global p_solup = 5.24 + 0.47 * rab
 else
  p_solup = 5.14 + 0.66 * rab
 end_if
 global p_sollo = 2.00 + pi
 local d_a1 = d_a + 1.
 local d_b1 = d_b + 1.
 global pdis0 = gp_near(0.0,0.0,0.0)
 local pdis1 = gp_near(d_a1,0.0,0.0)
 local pdis2 = gp_near(0.0,d_b1,0.0)
 global area = (gp_xpos(pdis1)+d_a)*(gp_ypos(pdis2)+d_b)*0.25
end
@p_cons
def p_load
 local pnt = gp_head
 local pload = 0.0
 local n = 0
 loop while pnt # null
 if gp_zpos(pnt) < eps then
  if gp_xpos(pnt) < ae then
   if gp_ypos(pnt) < be then
    pload = pload + gp_zfunbal(pnt)
    n = n + 1
   end_if
  end_if
 end_if
 pnt = gp_next(pnt)
 end_loop
 pload = - pload / (area * z_prop(zone_head,'cohesion'))
 p_load = pload
 global c_disp = gp_zdisp(pdis0) / d_a
 global p_errlo = 100. * (pload - p_sollo) / p_sollo
 NoZones =nzone
end
hist nstep 50
```

```
hist add fish @p_load
hist add fish @p_solup
hist add fish @p_sollo
hist add fish @c_disp
hist add unbal
save sslab
plot create view stress_displacement
plot add hist 1 2 3 vs 4
def const
 local bm = z_prop(zone_head,'bulk')
 local sm = z_prop(zone_head,'shear')
 global ny_Poisson = (3.*bm-2.*sm)/(6.*bm+2.*sm)
 global E_Young=3*bm*(1-2*ny_Poisson)
end
@const
plot create view shear_state
plot add zone colorby state
plot create view DispCont
plot set dip 90 dd 90 center 0 0 0
plot add cont disp plane behind
plot add axes
plot create view stresscontours
plot set dip 90 dd 90 center 0 0 0
plot add zonecont szz
plot add axes
def safetyfactor
;-------------------------------------------------------------------------------
; Obtains the inverse of Factor of Saftey [fstr] inside Flac3D Sub-domain
;
; str1: minor principal stress
; str3: major principal stress
; PHI: friction angle
; coh: cohesion
;-------------------------------------------------------------------------------
 command
  tab 1 name Fs_in_X_direction
 end_command
 pnt = zone_head
 loop while pnt # null
  loop igp (1,8)
   gpnt = z_gp(pnt,igp)
```

```
    mark = 0
    if gp_ypos(gpnt) = 0.0 then
     if gp_zpos(gpnt) = 0.0 then
      mark = mark+1
     end_if
    end_if
    if mark = 1 THEN
     str1 = (z_sig1(pnt))
     str3 = (z_sig3(pnt))
     PHI = z_prop(pnt,'friction')
     coh = z_prop(pnt,'cohesion')
     NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
     fstr=(str3 - str1)/(2*coh*sqrt(NPHI)+(str3)*(NPHI-1))
     radd= z_xcen(pnt)
     table(1,radd) = fstr
    end_if
  end_loop
  pnt = z_next(pnt)
end_loop
command
 tab 2 name Fs_in_Z_direction
end_command
pnt = zone_head
loop while pnt # null
 loop igp (1,8)
  gpnt = z_gp(pnt,igp)
  mark = 0
  if gp_xpos(gpnt) = 0.0 then
   if gp_ypos(gpnt) = 0.0 then
    mark = mark+1
   end_if
  end_if
  if mark = 1 THEN
   str1 = (z_sig1(pnt))
   str3 = (z_sig3(pnt))
   PHI = z_prop(pnt,'friction')
   coh = z_prop(pnt,'cohesion')
   NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
   fstr=(str3 - str1)/(2*coh*sqrt(NPHI)+(str3)*(NPHI-1))
   radd= z_zcen(pnt)
   table(2,radd) = fstr
```

```
    end_if
   end_loop
   pnt = z_next(pnt)
  end_loop
  command
   tab 3 name Fs_in_Y_direction
  end_command
  pnt = zone_head
  loop while pnt # null
   loop igp (1,8)
    gpnt = z_gp(pnt,igp)
    mark = 0
    if gp_xpos(gpnt) = 0.0 then
     if gp_zpos(gpnt) = 0.0 then
      mark = mark+1
     end_if
    end_if
    if mark = 1 then
     str1 = (z_sig1(pnt))
     str3 = (z_sig3(pnt))
     PHI = z_prop(pnt,'friction')
     coh = z_prop(pnt,'cohesion')
     NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
     fstr=(str3 - str1)/(2*coh*sqrt(NPHI)+(str3)*(NPHI-1))
     radd= z_ycen(pnt)
     table(3,radd) = fstr
    end_if
   end_loop
   pnt = z_next(pnt)
  end_loop
end
@test1
list @tim
plot create view Fs-in-X-direction
plot add table 1
plot create view Fs-in-Z-direction
plot add table 2
plot create view Fs-in-Y-direction
plot add table 3
```

```
;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------
; Define The Boundary Elements Incidences and Boundary Nodes Coordinates
;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------
def Element_nunmber
;-----------------------------------------------------------------
; Obtain The Boundary Elements Number NELM [1]
;-----------------------------------------------------------------
zpnt = zone_head
NELM=0
loop while zpnt # null
 mark = 0
 loop igp (1,8)
  gpnt = z_gp(zpnt,igp)
  distance=sqrt(gp_xpos(gpnt)^2+gp_ypos(gpnt)^2+gp_zpos(gpnt)^2)
  if distance >= len -0.001 then
   mark = mark+1
  end_if
 end_loop
 if mark = 4 THEN
  NELM = NELM+1
 end_if
 zpnt = z_next(zpnt)
end_loop
end
@Element_nunmber
def main_input
;-------------------------------------------------------
; Assign the basic information[2]
;-------------------------------------------------------
 array ar(1) arr(1) arrr(3,4) arrrr(1)
 filename1 = 'result.dat'
 oo = open(filename1,1,1)
 arr(1)='Project: Smooth square footing on Tresca material'
 oo=write(arr,1)
 global CD = 3    ; Cartesian dimension (2D & 3D)
 arr(1)='Cartesian_dimension ='+string(CD)
 oo=write(arr,1)
 ELD= CD-1        ; Boundary Element dimension
 DOF = 3          ; Degrees of freedom
```

```
TYSUB = 3        ; Type of Sub-domain (Finite BEM Sub-domain = 1, Infinite BEM Sub-domain = 2,Semi-
Infinite BEM Sub-domain = 3)
KelMind= 1       ; Method of Analysis for Semi-infinte Sub-domain or Type of Used Fundamental solutions(
Kelvin's = 0, Mindlin's = 1, Melan's = 2)
IF TYSUB = 1 THEN
 arr(1)='Finite Region'
 oo=write(arr,1)
ELSE
 IF TYSUB = 2 THEN
  arr(1)='Infinite Region'
  oo=write(arr,1)
 ELSE
  IF KelMind=0
   arr(1)='Semi-Infinite Region/ Kelvin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=1
   arr(1)='Semi-Infinite Region/Mindlin solution '
   oo=write(arr,1)
  END_IF
  IF KelMind=2
   arr(1)='Semi-Infinite Region/Melan solution '
   oo=write(arr,1)
  END_IF
 END_IF
END_IF
TYM = 2 ; Symmetry about yz = 1, Symmetry about yz and xz = 2, Symmetry about yz, xz and xy =3)
CASEOF TYM
 arr(1)='No symmetry'
 oo=write(arr,1)
 CASE 1
  arr(1)='Symmetry about y-z plane'
  oo=write(arr,1)
 CASE 2
  arr(1)='Symmetry about y-z and x-z planes'
  oo=write(arr,1)
 CASE 3
  arr(1)='Symmetry about all planes'
  oo=write(arr,1)
ENDCASE
SPS= 1
```

```
 ELTY=1 ; Element type
 ; Determine number of nodes per element NDEL
 IF CD = 2 THEN
  NDEL= 2
 ELSE
  NDEL= 4
 END_IF
 arr(1)=' Single Element Nodes number = '+ string(NDEL)
 oo=write(arr,1)
 ; Material Properties
 arr(1)=' Young modulus = '+string(E_Young)
 oo=write(arr,1)
 arr(1)=' Poisson ratio= '+string(ny_Poisson)
 oo=write(arr,1)
 iterations = 50 ; No. of iterations
 iterations1=iterations+1 ; Temp
 arr(1)=' Number of Iterations= '+string(iterations)
 oo=write(arr,1)
 relaxation=0.4
 arr(1)=' relaxation Parameter= '+string(relaxation)
 oo=write(arr,1)
 iteration_Tolerance = 0.0005 ; iteration tolerance should be higher if function NextiterFlacvelocity1 is
 used (0.002)
 arr(1)=' iteration Tolerance= '+string(iteration_Tolerance)
 oo=write(arr,1)
 Factorepsolon = 0.01
 VCD= 2*CD      ; No. of Stress Components at a Node
 DOFL= NDEL*DOF ; Total degrees of freedom of element
 INTE=24
 INTP = 3*INTE ; Number of internal points displacement and stress are required to be computed at
 NELI= 1
 NELF= NELM
 NDEL1= NDEL+1
 oo = close
end
@main_input
def Matrices_dimensions1
;-------------------------------------------------------------------------
; Define the Program Arrays Dimensions
;-------------------------------------------------------------------------
 array SHFUN(NDEL) DESHFUN(NDEL,ELD)   ; Shape function & Derivatives of shape function
```

```
  array CoordL(CD,NDEL) Coord(CD)        ; element coordinates & Cartesian coordinates
  array v_Normalize(CD) v1_xproduct(CD) v2_xproduct(CD) xproduct1(CD)
  array v_Outnormal(CD) v1_Outnormal(CD) v2_Outnormal(CD) ; Vector normal to point & Vectors in xsai,etta
  directions
  array Incd(NELM,NDEL) v_BIC(CD) Coord_BIC(CD)
  array BELU(NELM,DOFL) BELT(NELM,DOFL)
  array BELU_Flac(NELM,DOFL,iterations1) BELU_Mindlin(NELM,DOFL)
end
@Matrices_dimensions1
def Temporaryarray
;---------------------------------------------------------------------
; Obtain Boundary Elements Temporary Incidences Array[3]
;---------------------------------------------------------------------
 Array BEL(NELM)
 Array IncdTemp(NELM,NDEL)
 zpnt = zone_head
 BEN=1
 loop while zpnt # null
  mark = 0
  N=0
  loop igp (1,8)
   gpnt = z_gp(zpnt,igp)
   distance=sqrt(gp_xpos(gpnt)^2+gp_ypos(gpnt)^2+gp_zpos(gpnt)^2)
   if BEN <= NELM Then
    if distance >= len-0.001 then
     N=N+1
     IncdTemp(BEN,N) = gp_id(gpnt)
     TNODTempf=Max(TNODTempf,IncdTemp(BEN,N))
     mark =mark+1
    end_if
   end_if
  end_loop
  if mark = 4 THEN
   BEL(BEN)=z_id(zpnt)
   NELMTemp=Max(NELMTemp,BEL(BEN))
   BEN = BEN + 1
  end_if
  zpnt = z_next(zpnt)
 end_loop
end
@Temporaryarray
```

```
 def BoundaryArrayFlac
 ;-------------------------------------------------------------------------
 ; Obtain Boundary Nodes Temporary Coordinates Array [4]
 ;-------------------------------------------------------------------------
  Array IncdFlac(NELM,NDEL1)
  Array IncdFlacT(NELM,NDEL1)
  Array NCCrd_Flac(CD,TNODTempf)
  Array IncdLFlac(NDEL)
  loop BEN(1,NELM)
   IncdFlac(BEN,1)=BEL(BEN)
   loop m(2,NDEL1)
    IncdFlac(BEN,m)=IncdTemp(BEN,m-1)
    IncdFlacT(BEN,m)=IncdFlac(BEN,m)
   end_loop
   Temporary1=IncdFlac(BEN,4)
   Temporary2=IncdFlac(BEN,5)
   IncdFlac(BEN,4)=Temporary2
   IncdFlac(BEN,5)=Temporary1
   IncdFlacT(BEN,4)=Temporary2
   IncdFlacT(BEN,5)=Temporary1
  end_loop
  loop m(1,NELM)
   Loop n(2,NDEL1)
    IF IncdFlac(m,n)#0 THEN
     gpnt=find_gp(IncdFlac(m,n))
     NCCrd_Flac(1,IncdFlac(m,n))=gp_xpos(gpnt)
     NCCrd_Flac(2,IncdFlac(m,n))=gp_ypos(gpnt)
     NCCrd_Flac(3,IncdFlac(m,n))=gp_zpos(gpnt)
    END_IF
   end_loop
  end_loop
 end
 @BoundaryArrayFlac
def Shape_Function
;-------------------------------------------------------------------
; Obtain shape functions
;-------------------------------------------------------------------
 loop n (1,NDEL)
  SHFUN(n)=0
 end_loop
 IF CD = 2 THEN
```

```
   SHFUN(1)= 0.5*(1.0 - xsai)
   SHFUN(2)= 0.5*(1.0 + xsai)
  ELSE
   SHFUN(1)= 0.25*(1.0-xsai)*(1.0-etta)
   SHFUN(2)= 0.25*(1.0+xsai)*(1.0-etta)
   SHFUN(3)= 0.25*(1.0+xsai)*(1.0+etta)
   SHFUN(4)= 0.25*(1.0-xsai)*(1.0+etta)
  END_IF
END
def Derive_Shape_Function
;-------------------------------------------------------------------
; Obtain Derivatives of shape functions
;-------------------------------------------------------------------
 loop m (1,ELD)
  loop n (1,NDEL)
   DESHFUN(n,m) =0
  end_loop
 end_loop
 IF CD=2 THEN
  DESHFUN(1,1)= -0.5
  DESHFUN(2,1)= 0.5
 ELSE
  DESHFUN(1,1)= -0.25*(1.0-etta)
  DESHFUN(1,2)= -0.25*(1.0-xsai)
  DESHFUN(2,1)= 0.25*(1.0-etta)
  DESHFUN(2,2)= -0.25*(1.0+xsai)
  DESHFUN(3,1)= 0.25*(1.0+etta)
  DESHFUN(3,2)= 0.25*(1.0+xsai)
  DESHFUN(4,1)= -0.25*(1.0+etta)
  DESHFUN(4,2)= 0.25*(1.0-xsai)
 END_IF
END
def Coordinate
;-----------------------------------------------------
; Obtain Cartesian coordinates
;-----------------------------------------------------
 LOOP n (1,CD)
  Coord(n)=0
 END_LOOP
 LOOP i (1,CD)
  LOOP j(1,NDEL)
```

```
    Coord(i)= Coord(i)+CoordL(i,j)*SHFUN(j)
   END_LOOP
  END_LOOP
END
def Normalize
;---------------------------------------
; Normalise vector
;---------------------------------------
 IF CD=2 THEN
  M_Normalize= SQRT(v_Normalize(1)^2+v_Normalize(2)^2)
 ELSE
  M_Normalize= SQRT(v_Normalize(1)^2+v_Normalize(2)^2+v_Normalize(3)^2)
 END_IF
 IF M_Normalize = 0 THEN
  EXIT
 END_IF
 LOOP n (1,CD)
  v_Normalize(n)= v_Normalize(n)/M_Normalize
 END_LOOP
END
def x_product
;---------------------------------------
; x-product v1xv2
;---------------------------------------
 xproduct1(1)=V1_xproduct(2)*V2_xproduct(3)-V2_xproduct(2)*V1_xproduct(3)
 xproduct1(2)=V1_xproduct(3)*V2_xproduct(1)-V1_xproduct(1)*V2_xproduct(3)
 xproduct1(3)=V1_xproduct(1)*V2_xproduct(2)-V1_xproduct(2)*V2_xproduct(1)
END
def Outnormal
;-----------------------------------------------------------
; Obtains the outward normal vector
;-----------------------------------------------------------
 LOOP n (1,CD)
  v1_Outnormal(n) =0
  v2_Outnormal(n) =0
 END_LOOP
 xsai=xsai_Outnormal
 etta=etta_Outnormal
 Derive_Shape_Function
 LOOP i (1,CD)
  LOOP j(1,NDEL)
```

```
    V1_Outnormal(i)= V1_Outnormal(i)+CoordL(i,j)*DESHFUN(j,1)
    IF CD=3 THEN
     V2_Outnormal(i)= V2_Outnormal(i)+CoordL(i,j)*DESHFUN(j,2)
    END_IF
   END_LOOP
  END_LOOP
  LOOP n(1,CD) ; normal vector
   V1_xproduct(n)= v1_Outnormal(n)
   V2_xproduct(n)= v2_Outnormal(n)
  end_loop
  IF CD = 2 THEN
   v_Outnormal(1)= V1_xproduct(2)
   v_Outnormal(2)= -V1_xproduct(1)
  ELSE
   x_product
   LOOP n(1,CD)
    v_Outnormal(n)= xproduct1(n)
   end_loop
  END_IF
  LOOP n(1,CD) ; Normalise
   v_Normalize(n)=v_Outnormal(n)
  END_LOOP
  Normalize
  LOOP n(1,CD)
   v_Outnormal(n) = v_Normalize(n)
  END_LOOP
END
def BoundaryIncidences
;-------------------------------------------------------------------
; Obtain the Boundary Elements Outward Normal and
; Their Incidences Array [5]
;-------------------------------------------------------------------
  LOOP BEN(1,NELM)
   LOOP m(2,NDEL1)
    LOOP n (1,CD)
     CoordL(n,m-1)= NCCrd_Flac(n,IncdFlac(BEN,m))
    END_LOOP
    IncdLFlac(m-1)= IncdFlac(BEN,m)
   END_LOOP
   LOOP SC (1,CD)
    v_Outnormal(SC) =0
```

```
  END_LOOP
  xsai_Outnormal=0
  etta_Outnormal=0
  Outnormal
  LOOP SC (1,CD)
   v_BIC(SC)= v_Outnormal(SC)
  END_LOOP
  Shape_Function
  Coordinate
  LOOP SC (1,CD)
   Coord_BIC(SC)= Coord(SC)
  END_LOOP
  Costh_BIC=0
  loop Sc (1,CD)
   Costh_BIC=Costh_BIC+ V_BIC(Sc)*Coord_BIC(Sc)
  end_loop
  IF CD = 3 THEN
   Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2+Coord_BIC(3)^2)
  ELSE
   Costh=sqrt(Coord_BIC(1)^2+Coord_BIC(2)^2)
  END_IF
  Costh_BIC=Costh_BIC/Costh
  IF CD = 3 THEN
   IF Costh_BIC > 0 Then
    IncdFlac(BEN,2)= IncdFlacT(BEN,5)
    IncdFlac(BEN,3)= IncdFlacT(BEN,4)
    IncdFlac(BEN,4)= IncdFlacT(BEN,3)
    IncdFlac(BEN,5)= IncdFlacT(BEN,2)
   END_IF
  END_IF
  IF CD = 2 THEN
   IF Costh_BIC > 0 Then
    IncdFlac(BEN,2)= IncdFlacT(BEN,3)
    IncdFlac(BEN,3)= IncdFlacT(BEN,2)
   END_IF
  END_IF
 END_LOOP
 LOOP BEN(1,NELM)
  LOOP m(1,NDEL1)
   IncdFlacT(BEN,m)= IncdFlac(BEN,m)
  End_LOOP
```

```
 End_LOOP
 I=0
 LOOP BEN(1,NELM)
  LOOP m(2,NDEL1)
   I=I+1
   IncdFlac(BEN,m)= I
   IF BEN >1 THEN
    SECTION
     LOOP s(1,BEN-1)
      LOOP mm(2,NDEL1)
       IF IncdFlacT(s,mm)=IncdFlacT(BEN,m) THEN
        IncdFlac(BEN,m)=IncdFlac(s,mm)
        I=I-1
        EXIT SECTION
       END_IF
      End_LOOP
     End_LOOP
    END_SECTION
   END_IF
  End_LOOP
 End_LOOP
 loop BEN(1,NELM)
  loop m(2,NDEL1)
   Incd(BEN,m-1)=IncdFlac(BEN,m)
   TNOD=Max(TNOD,Incd(BEN,m-1))
  end_loop
 end_loop
 TNODDOF=TNOD*DOF
 oo = open(filename1,2,1)
 arr(1)='Number of Nodes of The System= '+string(TNOD)
 oo=write(arr,1)
 arr(1)='Number of Elements of The System= '+string(NELM)
 oo=write(arr,1)
 oo=close
end
 @BoundaryIncidences
 def NodesCoordinates
 ;-------------------------------------------------------------------------
 ; Compute Boundary Nodes Coordinates [6]
 ;-------------------------------------------------------------------------
  array NCCrd(CD,TNOD) ; Boundary Node co-ordinates
```

```
 array NCCrd_Mindlin(CD,TNOD)
 LOOP BEN(1,NELM)
  LOOP m(2,NDEL1)
   LOOP n (1,CD)
    NCCrd(n,Incd(BEN,m-1))= NCCrd_Flac(n,IncdFlacT(BEN,m))
    if abs(NCCrd(n,Incd(BEN,m-1))) < 10e-10 then
     NCCrd(n,Incd(BEN,m-1)) = 0.0
    end_if
   END_LOOP
  END_LOOP
 END_LOOP
 LOOP m(1,TNOD)
  IF TYSUB = 3 THEN
   IF CD=3 THEN
    NCCrd_Mindlin(2,m) = NCCrd(1,m)
    NCCrd_Mindlin(1,m) = NCCrd(3,m)
    NCCrd_Mindlin(3,m) = NCCrd(2,m)
   ELSE
    NCCrd_Mindlin(2,m) = NCCrd(1,m)
    NCCrd_Mindlin(1,m) = NCCrd(2,m)
   END_IF
  END_IF
 END_LOOP
end
@NodesCoordinates
def Matrices_dimensions2
;-------------------------------------------------------------------------
; Define the Program Matrices Dimensions
;-------------------------------------------------------------------------
 array BELf_Flac(TNODDOF)BELf_FlacMindlin(TNODDOF)
 array NCCrd_INT(INTP,CD) StressBE(NELM,NDEL,VCD)
 array U_INT(INTP,CD) Stress_INT(INTP,VCD)
 array epselontolEL(iterations1)
 array PSt(CD)   ; Principal Stress
end
@Matrices_dimensions2
def Input_INT
;-------------------------------------------------------------------------
; Reads Internal Points Coordinates and Writes Boundary Nodes Coordinates
;-------------------------------------------------------------------------
 nt=0
```

```
loop j (1,INTE)
 nt=nt+1
 NCCrd_INT(nt,1)= (1.0+(j/10.0))*len*sin(pi/(8*in_size))
 NCCrd_INT(nt,2)= (1.0+(j/10.0))*len*cos(pi/(8*in_size))
 NCCrd_INT(nt,3)= NCCrd_INT(nt,1)
endloop
INTP1=nt
loop j (1,INTE)
 nt=nt+1
 NCCrd_INT(nt,1)= (1.0+(j/25.0))*len*cos(pi/(8*in_size))
 NCCrd_INT(nt,2)= (1.0+(j/25.0))*len*sin(pi/(8*in_size))
 NCCrd_INT(nt,3)= NCCrd_INT(nt,2)
endloop
INTP2=nt
loop j (1,INTE)
 nt=nt+1
 NCCrd_INT(nt,1)= (1.0+(j/10.0))*len*sin(pi/(8*in_size))
 NCCrd_INT(nt,2)= NCCrd_INT(nt,1)
 NCCrd_INT(nt,3)= (1.0+(j/10.0))*len*cos(pi/(8*in_size))
endloop
oo = open(filename1,2,1)
arr(1)='Nodes Coordinates:'
oo=write(arr,1)
loop m (1,TNOD)
 arr(1)=' '
 loop n (1,CD)
  IF TYSUB # 3 THEN
   arr(1) =string(arr(1))+' '+ string(NCCrd(n,m))
  ELSE
   arr(1) =string(arr(1))+' '+ string(NCCrd_Mindlin(n,m))
  ENDIF
 end_loop
 ar(1)='Node No.'+string(m)
 oo = write(ar,1)
 oo = write(arr,1)
end_loop
arr(1)='Elements Incidences:'
oo=write(arr,1)
loop m (1,NELM)
 arr(1)=' '
 loop n (1,NDEL)
```

```
    arr(1) =string(arr(1))+' '+string(Incd(m,n))
   end_loop
   ar(1)='ELEMENT No.'+string(m)
   oo = write(ar,1)
   oo = write(arr,1)
  end_loop
  arr(1)='Coordinates of points in unbounded domain:'
  oo=write(arr,1)
  loop n(1,INTP)
   arr(1)=' '
   loop m(1,CD)
    arr(1) =string(arr(1))+' '+string(NCCrd_INT(n,m))
   end_loop
   ar(1)='point No.'+string(n)
   oo = write(ar,1)
   oo = write(arr,1)
  end_loop
  oo=close
END
def writing1
;---------------------------------------
; Write Some of FISH Functions
;---------------------------------------
oo = open(filename1,2,1)
arr(1)=' '
oo = write(arr,1)
loop BEN (1,NELM)
 ar(1)='RESULTS OF ELEMENT No.'+string(BEN)
 oo = write(ar,1)
 arr(1)= 'Displacement u:'
 oo=write(arr,1)
 arr(1)=' '
 arrrr(1)=' '
 loop n(1,DOFL)
  IF TYSUB = 3 THEN
   arr(1) =string(arr(1))+' '+string(BELU_Mindlin(BEN,n))
  else
   arr(1) =string(arr(1))+' '+string(BELU(BEN,n))
  ENDIF
  arrrr(1) =string(arrrr(1))+' '+string(BELT(BEN,n))
 end_loop
```

```
  oo = write(arr,1)
  arr(1)= 'Traction t:'
  oo=write(arr,1)
  oo = write(arrrr,1)
 end_loop
 oo=close
 end
 def writing2
 ;-------------------------------------
 ; Write Some of FISH Functions
 ;-------------------------------------
  oo = open(filename1,2,1)
  ar(1)='BELU_Flac at iteration NO.='+string(iter)+' / epsolon='+string(epselontolEL(iter))
  oo = write(ar,1)
  arr(1)=' '
  oo = write(arr,1)
  loop BEN (1,NELM)
   ar(1)='RESULTS OF ELEMENT No.'+string(BEN)
   oo = write(ar,1)
   arr(1)= 'BELU_Flac:'
   oo=write(arr,1)
   arr(1)=' '
   arrrr(1)=' '
   loop n(1,DOFL)
    arr(1) =string(arr(1))+' '+string(BELU_Flac(BEN,n,iter))
    arrrr(1) =string(arrrr(1))+' '+string(BELU(BEN,n))
   end_loop
   oo = write(arr,1)
   arr(1)= 'BELU:'
   oo=write(arr,1)
   oo = write(arrrr,1)
  end_loop
  oo=close
 end
 def applyFlacvelocity
 ;-----------------------------------------------------------------
 ; Apply velocity over the Flac3D side of the interface
 ;-----------------------------------------------------------------
  loop BEN(1,NELM)
   Loop m(2,NDEL1)
    if IncdFlacT(BEN,m)#0 then
```

```
      gpnt=find_gp(IncdFlacT(BEN,m))
      idv = IncdFlacT(BEN,m)
      if CD=3 then
       vx = BELU_Flac(BEN,3*(m-1)-2,iter)/STEPNUMBER
       vy = BELU_Flac(BEN,3*(m-1)-1,iter)/STEPNUMBER
       vz = BELU_Flac(BEN,3*(m-1),iter)/STEPNUMBER
       COMMAND
        Apply xv @vx range id @idv
        Apply yv @vy range id @idv
        Apply zv @vz range id @idv
       END_COMMAND
      else
       gpnt1=find_gp(IncdFlacT1(BEN,m))
       idv1 = IncdFlacT1(BEN,m)
       vx = BELU_Flac(BEN,2*(m-1)-1,iter)/STEPNUMBER
       vz = BELU_Flac(BEN,2*(m-1),iter)/STEPNUMBER
       COMMAND
        Apply xv @vx range id @idv
        Apply zv @vz range id @idv
        Apply xv @vx range id @idv1
        Apply zv @vz range id @idv1
       END_COMMAND
       if SPS=1 then
        COMMAND
         Apply yv 0 range id @idv
         Apply yv 0 range id @idv1
        END_COMMAND
       end_if
      end_if
     end_if
    end_loop
   end_loop
end
def NextiterFlacvelocity1
;-------------------------------------------------------------------------
; Obtain the interface nodal velocity vector for the next iteraion
; [First coupling method]
;-------------------------------------------------------------------------
 if TYSUB = 3 then
  loop BEN(1,NELM)
   loop m(1,NDEL)
```

```
      IF CD=3 THEN
       BELU(BEN,3*m-2)= BELU_Mindlin(BEN,3*m-1)
       BELU(BEN,3*m-1)= BELU_Mindlin(BEN,3*m)
       BELU(BEN,3*m)= BELU_Mindlin(BEN,3*m-2)
      ELSE
       BELU(BEN,2*m-1)= BELU_Mindlin(BEN,2*m)
       BELU(BEN,2*m)= BELU_Mindlin(BEN,2*m-1)
      ENDIF
     endloop
    endloop
   endif
   loop BEN(1,NELM)
    Loop m(1,DOFL)
     BELU_Flac(BEN,m,iter+1)=((1-relaxation)*BELU_Flac(BEN,m,iter)+relaxation*BELU(BEN,m))
    end_loop
   end_loop
   nominator= 0
   denominator= 0
   loop BEN(1,NELM)
    loop m(1,DOFL)
     nominator= nominator+(BELU_Flac(BEN,m,iter+1)-BELU_Flac(BEN,m,iter))*(BELU_Flac(BEN,m,iter+1)-
     BELU_Flac(BEN,m,iter))
     denominator=denominator+BELU_Flac(BEN,m,iter+1)*BELU_Flac(BEN,m,iter+1)
    end_loop
   end_loop
   epselontolEL(iter)=sqrt(nominator/denominator)
   IF iter > 1
   ;IF epselontolEL(iter) < iteration_Tolerance THEN
    IF epselontolEL(iter-1) < epselontolEL(iter) THEN
     iterating= 1
    END_IF
   END_IF
end
def NextiterFlacvelocity3
;-------------------------------------------------------------------------
; Obtain the interface nodal velocity vector for the next iteraion
; [First coupling method with different convergence condition]
;-------------------------------------------------------------------------
 if TYSUB = 3 then
  loop BEN(1,NELM)
   loop m(1,NDEL)
```

```
      IF CD=3 THEN
       BELU(BEN,3*m-2)= BELU_Mindlin(BEN,3*m-1)
       BELU(BEN,3*m-1)= BELU_Mindlin(BEN,3*m)
       BELU(BEN,3*m)= BELU_Mindlin(BEN,3*m-2)
      ELSE
       BELU(BEN,2*m-1)= BELU_Mindlin(BEN,2*m)
       BELU(BEN,2*m)= BELU_Mindlin(BEN,2*m-1)
      ENDIF
     endloop
    endloop
   endif
   loop BEN(1,NELM)
    loop m(1,DOFL)
     BELU_Flac(BEN,m,iter+1)=((1-relaxation)*BELU_Flac(BEN,m,iter)+relaxation*BELU(BEN,m))
    end_loop
   end_loop
   Tempo= 0
   loop BEN(1,NELM)
    loop m(1,DOFL)
     Tempo=Tempo+(BELU_Flac(BEN,m,iter)-BELU(BEN,m))*(BELU_Flac(BEN,m,iter)-BELU(BEN,m))
    end_loop
   end_loop
   epselontolEL(iter)=sqrt(Tempo)
   ERFPW = (1.0/iter)
   ERF = (epselontolEL(iter) / epselontolEL(1))^(ERFPW)
   IF iter > 1
    if epselontolEL(iter) < Factorepsolon * epselontolEL(1) THEN
     NIT =iter
    end_if
;  IF epselontolEL(iter) < iteration_Tolerance THEN
    IF epselontolEL(iter-1) < epselontolEL(iter) THEN
     iterating = 1
    END_IF
   END_IF
end
def ApplyFlacforce
;--------------------------------------------------------------------
; Obtain the interface nodal forces vector
;--------------------------------------------------------------------
 LOOP m(1,TNODDOF)
  BELf_Flac(m)=0
```

```
  END_LOOP
  loop BEN(1,NELM)
   Loop m(2,NDEL1)
    IF IncdFlacT(BEN,m)#0 THEN
     gpnt=find_gp(IncdFlacT(BEN,m))
     IF CD=3 Then
      BELf_Flac(3*Incd(BEN,m-1)-2)=gp_xfunbal(gpnt)+gp_xfapp(gpnt)
      BELf_Flac(3*Incd(BEN,m-1)-1)=gp_yfunbal(gpnt)+gp_yfapp(gpnt)
      BELf_Flac(3*Incd(BEN,m-1))=gp_zfunbal(gpnt)+gp_zfapp(gpnt)
     ELSE
      BELf_Flac(2*Incd(BEN,m-1)-1)=(2/len2)*(gp_xfunbal(gpnt)+gp_xfapp(gpnt))
      BELf_Flac(2*Incd(BEN,m-1))=(2/len2)*(gp_zfunbal(gpnt)+gp_zfapp(gpnt))
     ENDIF
    ENDIF
   end_loop
  end_loop
  IF TYSUB = 3 THEN
   LOOP m(1,TNOD)
    IF CD=3 Then
     BELf_FlacMindlin(3*m-1)= BELf_Flac(3*m-2)
     BELf_FlacMindlin(3*m) = BELf_Flac(3*m-1)
     BELf_FlacMindlin(3*m-2)= BELf_Flac(3*m)
    ELSE
     BELf_FlacMindlin(2*m)= BELf_Flac(2*m-1)
     BELf_FlacMindlin(2*m-1) = BELf_Flac(2*m)
    ENDIF
   END_LOOP
  ENDIF
 end
 def Convergence1
 ;-----------------------------------------------------------------
 ; First Method Coupling Iteration Loop
 ;-----------------------------------------------------------------
  loop BEN(1,NELM)
   Loop m(1,DOFL)
    BELU_Flac(BEN,m,1)=0 ; apply initial velocity on Flac sub-domain interface nodes
   end_loop
  end_loop
  SECTION
   loop iter(1,iterations)
    if iter = 1 then
```

```
 Command
  fix z range x -0.1 @ae y -0.1 @be z -0.1 0.1
 EndCommand
end_if
Command
 ini sxx 0.0 syy 0.0 szz 0.0 sxy 0.0 sxz 0.0 syz 0.0
 ini xvel 0.0 yvel 0.0 zvel 0.0
 ini xdis 0.0 ydis 0.0 zdis 0.0
 ini zvel 2.5e-5 range x -0.1 @ae y -0.1 @be z -0.1 0.1
 apply xv 0 range x -.001 .001
 apply yv 0 range y -.001 .001
 @applyFlacvelocity
 solve step @STEPNUMBER
EndCommand
ApplyFlacforce
IF iter =1 THEN
 Input_INT
ENDIF
msg = 'iter='+string(iter)
dum = out(msg)
msg = 'Please wait for the BEM-C++ intrinsic /example_PreProcessing/ to excute its computations '
dum = out(msg)
if TYSUB = 3
 Pre = example_PreProcessing(CD,DOF,TYSUB,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,KelMind,SPS,
 NCCrd_Mindlin,Incd,BELf_FlacMindlin,BELU_Mindlin,BELT)
else
 Pre = example_PreProcessing(CD,DOF,TYSUB,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,KelMind,SPS,
 NCCrd,Incd,BELf_Flac,BELU,BELT)
end_if
NextiterFlacvelocity3
msg = ' NIT = '+ string (NIT)
dum = out(msg)
msg =' ERF = ' + string (ERF)
dum = out(msg)
table(101,iter) = NIT
table(102,iter) = ERF
IF iterating = 1 THEN
 ;if iter =iterations then
 writing1
 writing2
 EXIT SECTION
```

```
    END_IF
   end_loop
  END_SECTION
end
@Convergence1
@test1
list @tim
list @p_sollo @p_load @p_solup
list @p_errlo
@safetyfactor
pause
def PostProcess
;-------------------------------------------------------------------------
; Obtain the stress and the displacement at chosen internal points
; and the stress on the boundary (interface)
;-------------------------------------------------------------------------
 msg = 'Please wait for the BEM-C++ intrinsinc /example_PostProcessing/ to excute its computations '
 dum = out(msg)
 if TYSUB = 3
  Post =
  example_PostProcessing(CD,DOF,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,NELI,NELF,INTP,SPS,TYSUB,KelMin
  d,NCCrd_Mindlin,Incd,BELU_Mindlin,BELT,NCCrd_INT,StressBE,U_INT,Stress_INT)
 else
  Post =
  example_PostProcessing(CD,DOF,ELTY,E_Young,ny_Poisson,TYM,NDEL,TNOD,NELM,NELI,NELF,INTP,SPS,TYSUB,KelMin
  d,NCCrd,Incd,BELU,BELT,NCCrd_INT,StressBE,U_INT,Stress_INT)
 end_if
 oo = open(filename1,2,1)
 arr(1)=' ------------------------------------------------'
 oo=write(arr,1)
 arr(1)=' Post-processed Results '
 oo=write(arr,1)
 arr(1)=' ------------------------------------------------'
 oo=write(arr,1)
 if NELI > 0 Then
  arr(1)='Results at Boundary Elements:'
  oo=write(arr,1)
  arr(1)= ' '
  oo=write(arr,1)
  loop BEN (1,NELF)
   loop Nodd (1,NDEL)
```

```
    arr(1)=' '
    arr(1) =string(arr(1))+'Element No.'+string(BEN)+' Global Node No. = '+string(Incd(BEN,Nodd))+' Local
    No./'+string(Nodd)+'/'
    oo = write(arr,1)
    ar(1)=' '
    loop m (1,VCD)
     ar(1) =string(ar(1))+' '+ string(StressBE(BEN,Nodd,m))
    end_loop
    oo = write(ar,1)
   end_loop
  end_loop
end_if
arr(1)=' '
oo = write(arr,1)
arr(1)='Internal Results:'
oo = write(arr,1)
arr(1)=' '
oo = write(arr,1)
loop intm(1,INTP)
 arr(1)='Coordinates'
 oo = write(arr,1)
 ar(1)=' '
 loop s (1,CD)
  ar(1) =string(ar(1))+' '+string(NCCrd_INT(intm,s))
 end_loop
 oo = write(ar,1)
 arr(1)=' displacement u: '
 oo = write(arr,1)
 ar(1)=' '
 loop ss (1,CD)
  ar(1) =string(ar(1))+' '+string(U_INT(intm,ss))
 end_loop
 oo = write(ar,1)
 arr(1)=' Stress: '
 oo = write(arr,1)
 ar(1)=' '
 loop sss (1,VCD)
  ar(1) =string(ar(1))+' '+string(Stress_INT(intm,sss))
 end_loop
 oo = write(ar,1)
end_loop
```

```
  oo=close
end
@PostProcess
@test1
list @tim
pause
def COLOUMB1
;-------------------------------------------------------------------------------------------
; returns inverse of Factor of Safety [fstr] based on Moher_Coloumb
; (or Tresca) Failure Criterion Value
;
; Note: The principal stress is computed in this function independent from FISH
;-------------------------------------------------------------------------------------------
 A_COB =1
 B_COB =-(Stress_INT(I1,1)+Stress_INT(I1,2)+Stress_INT(I1,3))
 C_COB=Stress_INT(I1,1)*Stress_INT(I1,2)+Stress_INT(I1,1)*Stress_INT(I1,3)+Stress_INT(I1,2)*Stress_INT(I1,
 3)-Stress_INT(I1,4)^2-Stress_INT(I1,5)^2-Stress_INT(I1,6)^2
 D_COB=-(Stress_INT(I1,1)*(Stress_INT(I1,2)*Stress_INT(I1,3)-Stress_INT(I1,5)^2)-
 Stress_INT(I1,4)*(Stress_INT(I1,4)*Stress_INT(I1,3)-
 Stress_INT(I1,5)*Stress_INT(I1,6))+Stress_INT(I1,6)*(Stress_INT(I1,4)*Stress_INT(I1,5)-
 Stress_INT(I1,2)*Stress_INT(I1,6)))
 F_COB=(((3*C_COB)/A_COB)-(B_COB^2/A_COB^2))/3
 G_COB=(((2*B_COB^3)/A_COB^3)-((9*B_COB*C_COB)/A_COB^2)+((27*D_COB)/A_COB))/27
 H_COB= (G_COB^2/4)+(F_COB^3/27)
 IF H_COB <= 0 THEN
  I_COB = sqrt((G_COB^2/4)-H_COB)
  J_COB = I_COB^0.3333
  K_COB = acos(-(G_COB/(2*I_COB)))
  L_COB =-J_COB
  M_COB =cos(K_COB/3)
  N_COB =sqrt(3.0)*sin(K_COB/3)
  P_COB =-(B_COB/3*A_COB)
  X1_COB=2*J_COB*cos(K_COB/3)-(B_COB/3*A_COB)
  X2_COB=L_COB*(M_COB+N_COB)+P_COB
  X3_COB=L_COB*(M_COB-N_COB)+P_COB
  MAX_X_COB=MAX(X1_COB,MAX(X2_COB,X3_COB))
  IF MAX_X_COB = X3_COB THEN
   PSt(3)= X3_COB
   PSt(2)= MAX(X1_COB,X2_COB)
   PSt(1)= MIN(X1_COB,X2_COB)
  ELSE
```

```
    IF MAX_X_COB = X2_COB THEN
     PSt(3)= X2_COB
     PSt(2)= MAX(X1_COB,X3_COB)
     PSt(1)= MIN(X1_COB,X3_COB)
    ENDIF
    IF MAX_X_COB = X1_COB THEN
     PSt(3)= X1_COB
     PSt(2)= MAX(X2_COB,X3_COB)
     PSt(1)= MIN(X2_COB,X3_COB)
    ENDIF
   ENDIF
   ELSE
    IF F_COB = 0.0 THEN
     IF G_COB =0.0 THEN
      IF H_COB =0.0 THEN
       PSt(1)= -(D_COB/A_COB)^0.3333
       PSt(2)=PSt(1)
       PSt(3)=PSt(1)
      END_IF
     END_IF
    END_IF
    IF H_COB > 0 THEN
     oo = open(filename1,2,1)
     ar(1)='Not ALL Roots Of Characteristic Equation Are Real'
     oo= write(ar,1)
     oo= close
     EXIT
    ENDIF
  ENDIF
  coh = z_prop(zone_head,'cohesion')
  PHI = z_prop(zone_head,'friction')
  Sigmat = z_prop(zone_head,'tension') ; Tensile Strength
  NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
  ap=sqrt(1+NPHI^2)+NPHI
  sigmap=Sigmat*NPHI-2*coh*sqrt(NPHI)
  hp=PSt(3)-Sigmat+ap*(PSt(1)-sigmap)
  IF hp <=0 THEN
   fs=PSt(1)- PSt(3)* NPHI+2*coh*sqrt(NPHI)
   SHEARESIST= 2*coh*sqrt(NPHI)-PSt(1)*(NPHI-1)
   ACTSHEAR = PSt(3)- PSt(1)
   IF fs <=0 THEN
```

```
    oo = open(filename1,2,1)
    ar(1)='Shear Failure'
    oo= write(ar,1)
    oo= close
   ELSE
    oo = open(filename1,2,1)
    ar(1)='Elastic Behaviour'
    oo= write(ar,1)
    oo= close
   ENDIF
   fstr= (PSt(3) - PSt(1))/(2*coh*sqrt(NPHI)+PSt(3)*(NPHI-1))
  ELSE
   ft=PSt(3)-Sigmat
   IF ft >=0 THEN
    oo = open(filename1,2,1)
    ar(1)='Tension Failure'
    oo= write(ar,1)
    oo= close
   ELSE
    oo = open(filename1,2,1)
    ar(1)='Elastic Behaviour'
    oo= write(ar,1)
    oo= close
   ENDIF
   fstr= PSt(3)/ Sigmat
  ENDIF
end
def COLOUMB2
;-------------------------------------------------------------------------------------
; returns inverse of Factor of Safety [fstr] based on Moher_Coloumb
; (or Tresca) Failure Criterion Value
;
; Note1: The principal stress is computed in this function by using FISH
; intrinsic: principal_stress
; Note2: The difference between the princpal stress values computed in
; functions COLOUMB1 and COLOUMB2 is minor
;-------------------------------------------------------------------------------------
 array PSt1(CD,CD) PStdir(CD)
 PSt1(1,1) = Stress_INT(I1,1)
 PSt1(2,2) = Stress_INT(I1,2)
 PSt1(3,3) = Stress_INT(I1,3)
```

```
PSt1(1,2) = Stress_INT(I1,4)
PSt1(2,1) = PSt1(1,2)
PSt1(1,3) = Stress_INT(I1,6)
PSt1(3,1) = PSt1(1,3)
PSt1(2,3) = Stress_INT(I1,5)
PSt1(3,2) = PSt1(2,3)
v3 = principal_stress(PSt1,PStdir)
PSt(1) = xcomp(v3)
PSt(3) = zcomp(v3)
coh = z_prop(zone_head,'cohesion')
PHI = z_prop(zone_head,'friction')
Sigmat = z_prop(zone_head,'tension')
NPHI=(1+sin(degrad*PHI))/(1-sin(degrad*PHI))
ap= sqrt(1+NPHI^2)+NPHI
sigmap= Sigmat*NPHI-2*coh*sqrt(NPHI)
hp= PSt(3)-Sigmat+ap*(PSt(1)-sigmap)
IF hp <=0 THEN
 fs= PSt(1)-PSt(3)* NPHI+2*coh*sqrt(NPHI)
 SHEARESIST= 2*coh*sqrt(NPHI)-PSt(1)*(NPHI-1)
 ACTSHEAR = PSt(3)- PSt(1)
 IF fs <=0 THEN
  oo = open(filename1,2,1)
  ar(1)='Shear Failure'
  oo= write(ar,1)
  oo= close
 ELSE
  oo = open(filename1,2,1)
  ar(1)='Elastic Behaviour'
  oo= write(ar,1)
  oo= close
 ENDIF
 fstr= (PSt(3) - PSt(1))/(2*coh*sqrt(NPHI)+PSt(3)*(NPHI-1))
ELSE
 ft=PSt(3)-Sigmat
 IF ft >=0 THEN
  oo = open(filename1,2,1)
  ar(1)='Tension Failure'
  oo= write(ar,1)
  oo= close
 ELSE
  oo = open(filename1,2,1)
```

```
    ar(1)='Elastic Behaviour'
    oo= write(ar,1)
    oo= close
   ENDIF
   fstr= PSt(3)/ Sigmat
  ENDIF
 end
 def Poststst
  ;-------------------------------------------------------------------------------
  ; Computes the inverse of Factor of Saftey [fstr] inside BEM Sub-domain
  ;-------------------------------------------------------------------------------
  command
   tab 4 name Fs_in_X_direction_Unbounded
   tab 5 name Fs_in_Z_direction_Unbounded
   tab 6 name Fs_in_Y_direction_Unbounded
  end_command
  loop i(1,INTP1)
   global I1 = i
   COLOUMB1
   radd= NCCrd_INT(i,2)
   table(4,radd) = fstr
   oo = open(filename1,2,1)
   arr(1)='Coordinates'
   oo = write(arr,1)
   ar(1)=' '
   loop s (1,CD)
    ar(1) =string(ar(1))+' '+string(NCCrd_INT(i,s))
   end_loop
   oo = write(ar,1)
   arr(1)=' Principal Stress: '
   oo = write(arr,1)
   ar(1)=' '
   loop s (1,CD)
    ar(1) =string(ar(1))+' '+string(PSt(s))
   end_loop
   oo = write(ar,1)
   oo = close
  end_loop
  loop i(INTP1+1,INTP2)
   global I1= i
   COLOUMB1
```

```
  radd= NCCrd_INT(i,1)
  table(5,radd) = fstr
  oo = open(filename1,2,1)
  arr(1)='Coordinates'
  oo = write(arr,1)
  ar(1)=' '
  loop s (1,CD)
   ar(1) =string(ar(1))+' '+string(NCCrd_INT(i,s))
  end_loop
  oo = write(ar,1)
  arr(1)=' Principal Stress: '
  oo = write(arr,1)
  ar(1)=' '
  loop s (1,CD)
   ar(1) =string(ar(1))+' '+string(PSt(s))
  end_loop
  oo = write(ar,1)
  oo = close
end_loop
loop i(INTP2+1,INTP)
 global I1=i
 COLOUMB1
 radd= NCCrd_INT(i,3)
 table(6,radd) = fstr
 oo = open(filename1,2,1)
 arr(1)='Coordinates'
 oo = write(arr,1)
 ar(1)=' '
 loop s (1,CD)
  ar(1) =string(ar(1))+' '+string(NCCrd_INT(i,s))
 end_loop
 oo = write(ar,1)
 arr(1)=' Principal Stress: '
 oo = write(arr,1)
 ar(1)=' '
 loop s (1,CD)
  ar(1) =string(ar(1))+' '+string(PSt(s))
 end_loop
 oo = write(ar,1)
 oo = close
end_loop
```

```
end
@Poststst
list tab 1
list tab 2
list tab 3
list tab 4
list tab 5
list tab 6
pause
plot create view Fs-XUD
plot add table 4
plot create view Fs-ZUD
plot add table 5
plot create view Fs-YUD
plot add table 6
pause
plot create view Fs-XBoth
plot add table 1, 4
plot create view Fs_ZBoth
plot add table 2, 5
plot create view Fs-YBoth
plot add table 3, 6
pause
```